The Dissertation Committee for Thomas Finnian O'Leary-Roseberry certifies that this is the approved version of the following dissertation:

Efficient and Dimension Independent Methods for Neural Network Surrogate Construction and Training

Committee:

Omar Ghattas, Supervisor

Patrick Heimbach, Co-Supervisor

George Biros

J. Tinsley Oden

Karen Willcox

Efficient and Dimension Independent Methods for Neural Network Surrogate Construction and Training

by

Thomas Finnian O'Leary-Roseberry

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2020

Dedicated to my family: Jim, Susan, Nate and Nora.

And the memory of my grandfather, Gar.

Acknowledgments

I thank my advisors Omar Ghattas and Patrick Heimbach, they supported me in pursuing problems that were challenging and interesting to me. I thank Professor Oden for convincing me in 2015 to join what has since been rightly named the Oden Institute.

I have greatly benefited from the intellectual environment at the Institute; I learned the most in long discussions with colleagues about optimization, solvers, approximations and other topics over the years. Within the CCGO, I benefited from mentorship from Peng Chen and Umberto Villa, and from many long conversations about math and other topics with Nick Alger and Josh Chen. Conversations I had with Professor Babuška, Professor Biros, Craig Michoski, Professor Oden and Professor Willcox helped me improve my research ideas.

At the University of Wisconsin–Madison, I acknowledge the mentorship I received from Riccardo Bonazza, Walter Drugan, Frank Rooney and Betsy Stovall, who all helped me decide to pursue a PhD.

I credit Umberto Villa, Peng Chen and Omar Ghattas for their help in my work on dimension reduced surrogates for parametric maps, for which they will be co-authors. I credit Nick Alger and Omar Ghattas for their collaboration on second order methods for stochastic nonconvex optimization, for which they are co-authors.

This document has benefited from comments and edits from Nick Alger, Omar Ghattas, Umberto Villa and Anna Yesypenko. I especially thank Anna for her extensive edits and comments.

Efficient and Dimension Independent Methods for Neural Network Surrogate Construction and Training

Publication No. _____

Thomas Finnian O'Leary-Roseberry, Ph.D.
The University of Texas at Austin, 2020

Supervisors: Omar Ghattas
Patrick Heimbach

In this dissertation I investigate how to efficiently construct neural network surrogates for parametric maps defined by PDEs, and how to use second order information to improve solutions to the related neural network training problem.

Many-query problems arising in scientific applications (such as optimization, uncertainty quantification and inference problems) require evaluation of an input output mapping parametrized by a high dimensional nonlinear PDE model. The cost of these evaluations makes solution using the model prohibitive, and efficient accurate surrogates are the key to solving these problems in practice. In this work I investigate neural network surrogates that use model information to detect informed subspaces of the input and output where the parametric map can be represented efficiently. These compact representations require relatively few data to train and outperform conventional data-driven approaches which require large training data sets.

Once a neural network is designed, training is a major issue. One seeks to find optimal weights for a neural network that generalize to data not seen during training. In this work I investigate how second order information can be efficiently exploited to design optimizers that have fast convergence and good generalization properties. These optimizers are shown to outperform conventional methods in numerical experiments.

Table of Contents

Acknowledgments **iv**

Abstract **v**

Chapter 1. Introduction **1**
1.1 Surrogates for Parametric Maps 1
1.2 Newton Methods for Stochastic Nonconvex Optimization . . . 4
1.3 General Remarks 8

Chapter 2. Model Based Neural Network Architecture Selection **9**
2.1 Introduction 9
2.1.1 Formal Definition of the Mapping 15
2.1.2 Input Projection: Finding model structure with first derivatives 17
2.1.3 Output Projection: Proper Orthogonal Decomposition . 19
2.1.4 Input-Output Error Bound for Optimal Ridge Function 21
2.2 Using Model Structure to Aid Architecture Design 23
2.3 Scalable Computation of the Input and Output Projectors . . 26
2.3.1 The action of ∇q and ∇q^T using adjoints 27
2.3.2 Approximation with randomized linear algebra 28
2.4 Numerical Experiments 29
2.4.1 Model for uncertain parameter m 32
2.4.2 Ice Sheet Model 32
2.4.3 Convection Reaction Diffusion Problem 40
2.5 Conclusion 56

Chapter 3. Newton Methods for Stochastic Nonconvex Optimization **60**
3.1 Background 68
3.2 Noise in stochastic indefinite Hessians 70
3.3 Inexact Newton Methods 76
3.4 Inexact Newton-Krylov Methods 78
3.4.1 Local convergence rates 79
3.4.2 Superior Approximation for Clustered Eigenvalues . . . 80
3.5 Low Rank Saddle Free Newton 82
3.5.1 Low Rank vs. Krylov 84
3.5.2 Scalable approximation and efficient inversion using randomized methods 85
3.5.3 Low Rank Saddle Free Newton Algorithm 87

3.5.4 Local convergence for the stochastic low rank Newton method . . . 89
3.5.5 Comparing costs: gradient vs. Hessian . . . 90
3.6 Numerical Experiments . . . 92
3.6.1 MNIST Classification . . . 93
Comparison of methods using globalization . . . 95
Comparison of methods using fixed steps . . . 98
3.6.2 CIFAR10 Convolutional autoencoder . . . 106
3.6.3 Parametric Map Regression . . . 114
3.6.4 Software . . . 118
3.7 Conclusion . . . 118
3.8 Convergence Bounds . . . 122
3.8.1 Convergence of Subsampled Inexact Newton Methods with Eisenstat-Walker . . . 122
3.8.2 Convergence of Inexact Newton Krylov Methods . . . 124
3.8.3 Convergence of Low Rank Newton Method . . . 128

Chapter 4. Conclusions **133**

Appendix **139**

Appendix 1. Ill-Posedness of Neural Network Training **140**
1.1 Introduction . . . 140
1.1.1 Notation and Definitions . . . 143
1.2 Stationary Points of Shallow Dense Network . . . 144
1.2.1 Zero misfit minima . . . 145
1.2.2 Strict Saddle Points and Spurious Local Minima. . . . 148
1.3 Extension to Deep Networks . . . 151
1.4 Addressing ill-posedness with regularization . . . 153
1.5 Conclusion . . . 154
1.6 Shallow Dense Neural Network Derivations . . . 154
1.6.1 Derivation of gradient . . . 154
1.6.2 Derivation of Hessian . . . 157
1.7 Deep Dense Neural Network Gradient Derivation . . . 159

Appendix 2. Scalable Methods for Learning Derivative Structure **161**
2.1 Efficient Derivative Training . . . 162
2.2 Approximation of derivative subspace . . . 164

Appendix 3. Miscellaneous Proofs **166**
3.1 Polynomials and Krylov Spaces . . . 166

Bibliography **169**

Chapter 1

Introduction

This thesis has two main thrusts:

1. Efficient methods for construction of discretization dimension independent neural network surrogates for parametric mappings using model structure.

2. Efficient matrix-free Newton methods for solving stochastic nonconvex optimization problems, including problems that arise in neural network training.

The two thrusts are connected, but can be taken entirely separately from each other.

1.1 Surrogates for Parametric Maps

I am interested in constructing neural network surrogates for settings in which an expensive, nonlinear parametric mapping needs to be queried many times. Typically the parametric map involves solution of a partial differential equation (PDE). For a given model parameter, m, one must solve the PDE for a state variable u, and the output quantity of interest, q, is a function of m, implicitly through the PDE solution for the state u, i.e. $q(m) = q(u(m))$. This many-query problem arises in many applications such as inverse problems, optimization, forward uncertainty quantification (UQ), optimal design, optimization under uncertainty and optimal experimental design, among others. In these settings the mapping often involves an expensive nonlinear high

dimensional model such as a PDE solution operator, which becomes the major computational bottleneck for the solution of the problem. Creating accurate and cheap surrogates for the parametric mapping makes the solution of these problems tractable in many cases where it would be otherwise impossible, if querying the high fidelity parametric mapping were required.

There has been significant interest recently in the application of neural networks to scientific problems, where parametric mappings typically arise. Neural networks have been very successful in data-driven applications such as computer vision and natural language processing. Modern computer architectural advances have made them more competitive at scale. The typical approaches used in data driven settings may not be appropriate in the model based setting for two main reasons.

1. In data driven applications, data are typically cheap and one can justify constructing neural network surrogates with very large configuration spaces, since there may be enough information in the very large data sets to infer all of the neural network weights without overfitting. In the setting of expensive physical models, generation of training data is a major expense; in some cases generating a single training data pair may take days to weeks.

2. Data may not have any known mathematical structure a priori for typical data driven applications (e.g. image classification etc.). In the setting of parametric maps, the input-output map has meaningful mathematical structure that can be exposed using the model.

Model based structure can be used to construct efficient neural network surrogates that only attempt to learn the input-output mapping in informed subspaces of the input and output. Informed subspaces are subspaces that the parametric mapping can be well approximated by when the input or output representation is restricted to them; they are the subspaces that capture the key information about the mapping.

Parametric mappings often have low dimensional structure; the mapping often is representable in a low dimensional subspace of the output, and may be sensitive to the input parameter only in a low dimensional subspace. The dimension of these subspaces is referred to as the intrinsic dimension of the input-output mapping. A surrogate model can only hope to resolve information in these informed subspaces of the input and output. Attempting to resolve information outside of these subspaces may be noise that is specific to instances of training data. Learning these stochastic fluctuations in the data leads to overfitting.

The intrinsic dimensionality of a parametric mapping is often several orders of magnitude smaller than the discretization dimension in the model that is used to evaluate the mapping.

In this work I investigate strategies for constructing neural network surrogates that are independent of the discretization dimension. They use model information to detect informed subspaces of the input and output spaces and learn the mapping in these informed subspaces. Approaches for neural network construction that are dependent on the discretization dimension may lead to severely underdetermined training problems, with very large configuration spaces for the neural network and few training data to inform them.

As neural networks configuration spaces grow, and the neural networks become more difficult to train, the energy landscapes may be littered with saddle points and spurious local minima. They can be highly sensitive to hyperparameter tuning such as initial guesses, and need sophisticated regularization schemes to make the problem well posed without polluting key information.

By restricting the neural network to only infer dominant modes of the mapping, fewer training data are required to inform these dominant modes. The important information content for the parametric map lies in these dominant subspaces. Further, since the neural network is restricted to only the informed subspaces of the input and output spaces, this makes it robust to

resolving noisy modes of the data by construction. Architecting neural networks only in informed subspaces can help guard against overfitting, and can be seen as a sort of problem regularization. By making the configuration space for the neural network smaller, training and evaluation of the neural network become significantly cheaper. The resulting training problem is better posed than the data-driven approach. The resulting neural network training energy landscape may have a higher concentration of generalizable local minima than the underdetermined problem.

I test this strategy on two different parametric mappings based on PDEs. First a quasi-linear 2D convection-reaction-diffusion uncertain parameter to observable mapping, and second a nonlinear 3D Stokes ice flow parameter to observable mapping.

The numerical results show that identification of the dominant modes of the input and output subspace (and corresponding intrinsic dimension) can lead to efficient neural network surrogate construction. When the discretization dimension is not too big, the conventional approach of making a network based on the discretization dimension can result in reasonable solutions, but parsimonious neural networks parametrized in the dominant subspace of the input and output can obtain comparable and often better generalization for a much smaller configuration space. As the discretization dimension grows the conventional strategy performs increasingly poorly due to the increased difficult of the training problem, while the informed subspace method does well in all cases.

This is a way of implicitly imposing model structure, by restricting the representation of the surrogate to only subspaces of the input and output where the model informs the mapping.

1.2 Newton Methods for Stochastic Nonconvex Optimization

Stochastic nonconvex optimization problems arise in many important computing applications; a critical example is neural network training, for

which one wants to represent an input-output mapping of data pairs $x \mapsto y$ by a neural network function $f(w, x)$ parametrized by weights w such that $f(w, x) \approx y$. The data pair x, y are conceived of as samples from a joint probability distribution $\nu(x, y)$. The empirical risk minimization can be stated mathematically as

$$\min_w F(w) = \mathbb{E}_{x,y\sim\nu}[\ell(w; x, y)], \tag{1.2.1}$$

where ℓ is a "loss" function quantifying the error of the neural network approximation of the output data in some norm.

Stochastic nonconvex optimization problems are solved iteratively performing the weight update

$$w_{k+1} = w_k + \alpha_k p_k, \tag{1.2.2}$$

where w_k is the weight vector for the neural network at iteration k, p_k is a candidate search direction for which the model might improve, and α_k is the step length taken in the search direction. In practice p_k is computed based on derivative information of $F(w)$.

Stochastic nonconvex optimization problems are difficult for a number of reasons:

- Nonconvexity of the problem.
- Dimensionality d_W of the neural network configuration space.
- Dimensionality of the training data used.
- Ill-conditioning of the problem.
- Stochasticity inherent in subsampling schemes used to reduce computational cost.

Due to the nonconvexity of the problem, finding the exact solution of the problem (a global minimum) is NP hard; one instead has to settle for local minima. The nonconvex energy landscape may be littered with saddle points

and spurious local minima (so-called because they do not generalize well to unseen data). In practice one has to traverse a landscape seeking stationary points and hoping that they are not saddle points or spurious local minima.

Due to the dimensionality of neural network configuration spaces and training data sets, subsampled first order methods have gained a great deal of popularity. Subsampled methods use small batches of training data, rather than the entire set at each iteration to increase computational economy. Second order methods are mostly avoided, since approximation of second order information (formation and factorization of Hessian matrix) is considered prohibitive when d_W is large. For this reason most attention is paid to first order methods in large scale stochastic nonconvex optimization problems.

Due to ill-conditioning of the problem, first order methods are slow to converge. This creates significant opportunity for second order methods, which rescale the problem based on second order information to speed up convergence.

Due to stochasticity, information can change drastically from one iteration to the next, and a stochastic optimizer can overfit to stochastic variations that are specific to the training data used, but not the underlying input-output map (noise).

In this work I study efficient matrix-free Newton methods that take these issues into account, and can outperform stochastic first order methods in stochastic nonconvex optimization problems.

I begin by analyzing statistical properties of the spectra of stochastic inexact Hessian like operators. In this analysis, it is shown that for a Gaussian model of a stochastic inexact Hessian, approximation errors for subsampled eigenvalues (relative to the true eigenvalues) can exhibit unbounded variance in regions where eigenvalues of the true stochastic Hessian change sign.

This theoretical result is demonstrated numerically in examples; when the Hessian is highly indefinite the spectrum can exhibit very high variance amongst many different subsampled approximations of the Hessian spectrum.

Only the dominant modes of the Hessian do not exhibit lots of variance (noise).

I study stochastic Newton Krylov methods and show that while they have excellent convergence properties, and can efficiently approximate a Newton direction for a very small number of matrix vector products, they cannot avoid this key issue of noise in the stochastic Hessian spectrum.

This leads to a novel algorithm, low rank saddle free Newton (LRSFN), which only attempts to approximate the dominant, low noise modes of the Hessian operator.

This low rank Hessian approximation can be computed and inverted efficiently using randomized methods and the Sherman Morrison Woodbury formula. Moreover as in saddle free Newton (SFN) [41] I take the absolute values of the eigenvalues of the Hessian approximation, which helps facilitate fast escape from indefinite regions where the unmodified Hessian may direct iterates towards a saddle point. Hessian approximations that make use of only positive eigenvalues do not effectively exploit negative curvature information. Unlike the SFN method, LRSFN only attempts to resolve persistent modes of the Hessian matrix.

I show that when the Hessian has low rank the LRSFN algorithm can achieve fast convergence in the vicinity of local minima.

I study inexact Newton CG and LRSFN methods on various neural network training problems: image classification, image compression (autoencoders), and parametric map regression. LRSFN's performance is superior to other methods on these problems, generalizing better to unseen data.

Inexact Newton CG and LRSFN outperform first order methods in numerical experiments I conduct. They can speed up convergence by efficiently approximating Hessian information. LRSFN can target and resolve modes of the Hessian that are not dominated by noise which leads to better generalization.

The LRSFN algorithm can be scaled to very large problems, since the

main computational bottleneck (product of the Hessian with multiple random vectors) can be parallelized across the random vectors. This is a clear advantage for LRSFN over Krylov methods, which are inherently sequential. The LRSFN algorithm is a scalable method that can achieve fast convergence and find highly generalizable local minima by ignoring noisy information.

1.3 General Remarks

The unifying theme of this dissertation project, beyond neural networks, is that of dimension reduction / exploitation of low dimensional information in stochastic problems. The methods I propose use information in highly informed subspaces to design scalable algorithms that are robust to higher order modes that may be noisy.

In the neural network architecture thrust of the thesis, this is done by restricting the neural network input-output map to only informed subspaces of the input and output spaces. This led to efficient neural network surrogates that could be efficiently trained, quickly evaluated and generalized to unseen data better than conventional approaches for neural network surrogate construction.

In the neural network training section, this dimension reduction is done by leveraging spectral properties of Hessians showing up in stochastic nonconvex optimization problems. In some cases the rank decays quickly, making low rank a good approximation of the Hessian. In other cases the Hessian spectrum clusters which makes the use of Krylov methods beneficial. When the Hessian is indefinite, the persistent modes of the Hessian live in a small subspace of the configuration space. The orthogonal complement to this subspace is likely to be dominated by noise and should not be incorporated into an approximate Newton solve. This leads to the LRSFN algorithm, which only attempts to approximate the Hessian information in modes which are not dominated by variance. This method performs well experimentally.

Chapter 2

Model Based Neural Network Architecture Selection

2.1 Introduction

The goal of this chapter is to explore the construction of discretization dimension independent neural network surrogates for expensive parametric mappings arising from discretizations of PDE type models. Efficient accurate surrogates are needed in many applications where these maps need to be queried many times.

The typical setting involves an uncertain model parameter m, which is sampled from a probability distribution ν. The output quantity of interest is a function $q(m)$, which depends on m implicitly via the evaluation of an expensive model (such as a high dimensional PDE). In this setting, the evaluation of the map $m \mapsto q$ requires expensive nonlinear solution for a state variable u that depends on m, then the quantity of interest q is calculated from the solution of the PDE. An example of this would understanding how uncertainty in a basal sliding boundary condition for an ice sheet model effects a uncertainty in a quantity of interest such as observations of the ice flow velocity at points on the surface of the ice sheet, or the flow of the ice sheet into the ocean.

Many-query problems for high dimensional models are intractable if the high dimensional model is to be used for each query of the mapping $m \mapsto q$; in the case of large scale climate models, evaluations of the input-output mapping can take days to weeks. For this reason a lot of work is done to construct efficient and accurate surrogate approximations of the map. The goal of surrogate construction is to construct approximations of the map that

are accurate over the input parameter probability distribution ν, i.e. to find a surrogate f that is inexpensive to construct and evaluate, such that

$$\mathbb{E}_\nu[\|q - f\|^2] = \int \|f(m) - q(m)\|^2 d\nu(m) < \epsilon \tag{2.1.1}$$

for a suitable tolerance $\epsilon > 0$, in a suitable norm. Here $\mathbb{E}_\nu$ is the expectation with respect to ν, as defined in equation (2.1.1).

Understanding low dimensional structure as well as nonlinearity is a key problem in constructing suitable cheaper surrogate models. In recent years, neural networks have become a nonlinear approximator of choice in data-driven applications, such as computer vision and natural language processing. The use of neural networks as approximating surrogates in scientific applications has gained a lot of interest.

Neural networks offer a computational framework to handle both dimension reduction as well as nonlinearity. Encoder-decoder networks are a class of neural network architecture that represent a nonlinear lower dimensional representation of an input-output map; they are sometimes referred to as nonlinear principal component analysis (PCA)[71]. Such representations can be used to compress information in a mapping.

The typical approaches for data-driven approaches to neural network construction may not be optimal for the settings of model reduction for high dimensional nonlinear parametric maps. In data-driven applications, data are cheap and extremely large data sets used for surrogate trained can be easily afforded. On the contrary, in scientific applications the generation of training data is the main bottleneck, and constructing large training data sets is infeasible. In applications with large training data sets, one can justify training neural networks with extremely large configuration spaces, since there will be enough training data to infer these modes. In this setting it is acceptable to have neural network architectures that scale with the dimensionality of the data (i.e. the input data).

In scientific applications the input data dimension typically scales with the size of the discretization for the model. For many models the discretiza-

tion dimension will be on the order of millions or greater. If the configuration space for the neural network is to scale naïvely with the discretization dimension, a commensurately large training data corpus will be needed to infer the weights for the neural network model. For expensive models this is simply out of the question.

In this setting data-driven approaches for neural network surrogate construction can lead to highly underdetermined problems, where there are very few training data relative to the size of the neural network configuration space. When this happens, the neural network is very likely to learn stochastic variations in the training data, which leads to overfitting, and thus poor generalization to new data. Neural networks with large configuration spaces can be very difficult to train; the resulting energy landscapes may have a large number of saddle points or other spurious stationary points that hobble optimizers.

In scientific applications the mapping $m \mapsto q$ often has low dimensional structure that is independent of the discretization dimension of the problem. In this setting, the input-output map is informed in a low dimensional manifold, and the dimension of this manifold is referred to as the intrinsic dimension of the problem. This intrinsic dimension is often several orders of magnitude smaller than the discretization dimension [4, 5, 6, 11, 17, 19, 20, 21, 22, 23, 32, 38, 47, 64, 87, 105]. For such mappings, neural networks should be able to efficiently approximate the mapping with a configuration space that scales with the intrinsic dimension of the problem, not the discretization dimension.

A major difficulty in architecture selection is that it is not clear in general how the intrinsic dimensions of the input and output spaces should be selected. Genetic algorithms can be used to select neural network architectures [16, 124, 138], but are expensive since each proposed network needs to be trained in order to test its viability, so this amounts to solving many "guess and check" training problems.

In this work I investigate how model information can be used to detect intrinsic dimensionality, which can be used to construct parsimonious neural network surrogates. This work is motivated by results in reduced order models (ROMs) and other projection based models that use informed subspaces of the input or output space to efficiently represent the mappings.

The intrinsic dimensionality of the parametric mapping can be found by computing dominant subspaces of the input and output spaces. In particular active subspace methods detect dominant subspaces of the input for with the output is most sensitive [139, 37]. Proper orthogonal decomposition detects the dominant subspaces of the output that are spanned by the output quantity of interest [82, 107]. Active subspaces can be computed efficiently using adjoint methods and randomized linear algebra. Proper orthogonal decompositions can be computed efficiently using randomized linear algebra and training data that are already computed. Both subspaces can be computed adaptively until an error tolerance is met, to ensure that the dominant modes of the input-output mapping are computed efficiently.

In this work I advocate for the construction of neural network surrogates that are restricted to learning the representation of the input-output map between these dominant subspaces. This methodology creates neural network surrogates that have configuration space dimension parametrized by the intrinsic dimension of the input-output map, and are independent of the discretization dimension. These neural network surrogates have much smaller configuration space dimension than typical data-driven approaches, and require far fewer training data in principle. Moreover, by restricting the network to learning only the dominant modes of the input-output mapping, it is robust to learning stochastic variations in the data that lie in the complimentary subspaces. The important information content for the parametric map lie in these dominant subspaces. Restricting models to only resolving these modes serves as a form of regularization. These neural network surrogates are more generalizable than the overparametrized approaches which

will ultimately begin to learn stochastic variations in the data.

This work is related to other works that use model information for dimension reduction [11, 24, 28, 30, 31, 79]. Similar work regarding dimension reduced neural network approximations specifically has been pursued in traditional machine learning applications, where principal component analysis (PCA) of the data are used to find low dimensional representations of the input and output spaces [52]. This approach has been recently applied to dimension reduction for parametric PDE surrogates [14]. Using PCA on the input subspace is related to the Karhunen-Loève expansion of the input parameter, which does not approximate the dominant modes of the input that inform the output as well as active subspace which use incorporate information about the sensitivity of the output to the input.

The idea of using active subspace dimension reduction in neural network construction has been pursued for low dimensional linear problems [130]. Some work has been done to use proper orthogonal decomposition for a reduced basis for the neural network representation [50, 77, 127]. In these cases model information is used to reduce the output space, as is customary in the reduced order modeling community, where parametric surrogates are constructed to approximate a high dimensional full order model in a reduced basis.

In order to further reduce the configuration spaces for the neural network representation I study residual network layers that use low rank linear algebra, which have been briefly explored in traditional machine learning applications [10]. Each layer of the neural network involves a compact nonlinear perturbation of identity of the form:

$$x_{i+1} = x_i + W_{i1}\sigma_i(W_{i0}x_i). \tag{2.1.2}$$

The matrices $W_{i1} \in \mathbb{R}^{r\times r_i}$ and $W_{i0} \in \mathbb{R}^{r_i\times r}$ are low rank if $r_i \ll r$, where r is the intrinsic dimension of the problem. Maps of this form are sparse to represent and have nice properties.I term these architectures projected low

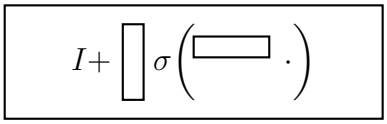


Figure 2.1: Low rank residual network layer schematic

rank residual neural networks (pLRRNs). This neural network architectural framework offers a parsimoniously parametrized, discretization dimension independent approximation for parametric input-output maps.

I test these neural network architectures against typical data-driven approaches on two different parametric map problems. The first is a 2D quasilinear convection diffusion PDE parameter to observable mapping, where the uncertain parameter is an uncertain volumetric forcing term. The second is a 3D nonlinear ice flow parameter to observable problem with an uncertain basal sliding coefficient. In these problems I find that the pLRRN neural networks are competitive with data-driven approaches when the discretization dimension is not too large, and significantly outperform these approaches when the discretization dimension increases. I find that identification of the intrinsic dimensionality of the map (as well as optimal basis representations for the input and output) is instrumental to constructing effective surrogates for parametric maps when the discretization dimension is very high. Moreover this approach allows for the possibility of a mesh independent surrogate, where optimal bases can be recomputed on different meshes. This can be thought of as a kind of transfer learning for parametric models [101, 129].

In total, I present a framework for automated neural network architecture design for parametric map surrogate construction. I investigate neural network architectures using low rank residual neural networks that allow for a great degree of dimension reduction beyond model based projectors. This approach allows for neural network surrogate construction that is independent of both the input and output dimensionality for the problem. These architectures were able to retain high level of accuracies for a sequence of refinements of a PDE problem, without having the configuration space grow at all. Nu-

merical results indicate that exposing intrinsic dimensionality may be critical for cheap neural network approximations of high dimensional maps arising from PDE based models. These models are parsimoniously parametrized, require fewer data for training than a conventional approach, are efficient to evaluate and could be of great use in high dimensional many-query settings.

2.1.1 Formal Definition of the Mapping

I begin by formally defining the setting of the parametric mapping. The target mappings come from infinite dimensional inference problems (typically involving PDEs) which can be broadly stated as follows. I begin by introducing the separable Banach spaces $\mathcal{U}, \mathcal{V}, \mathcal{M}$ where the state u, the adjoint v, and the uncertain model parameter m live, respectively. The uncertain parameter obeys a probability distribution ν, which may or may not be known in practice, but one seeks to approximate the mapping faithfully over the probability distribution. Given an instantiation of the uncertain model parameter m one seeks $u \in \mathcal{U}$ such that

$$\mathcal{R}(u, m) = 0 \qquad \text{in } \mathcal{V}', \tag{2.1.3}$$

where $\mathcal{R}(\cdot, m) : U \to \mathcal{V}'$ denotes a nonlinear mapping from $\mathcal{U}$ to the dual of $\mathcal{V}$. Equation (2.1.3) represents the **strong form** of the **state equation**. I define the associated weak form of the state equation via duality pairing: find $u \in \mathcal{U}$ such that

$$\mathbf{r}(u, m, v) = \langle v, \mathcal{R}(u, m) \rangle_{\mathcal{V}, \mathcal{V}'} = 0 \qquad \forall v \in \mathcal{V}. \tag{2.1.4}$$

Given a solution to the state equation, the quantity of interest is then a finite dimensional vector valued, continuous (and typically differentiable) function of the state $u \in \mathcal{U}$,

$$\mathbf{q} : \mathcal{U} \to \mathbb{R}^{d_Q}. \tag{2.1.5}$$

As stated, the quantity of interest depends on the uncertain parameter m implicitly through the solution of the state equation, however the framework can be easily extended to accommodate explicit dependence.

The model parameter m and state u live in infinite dimensional separable Banach spaces. The state variable u typically corresponds to a solution to a PDE and the parameter m is another infinite dimensional field that parametrizes the PDE. In practice the PDE model is discretized and solved on a computer. The infinite dimensional Banach spaces $\mathcal{U}, \mathcal{V}, \mathcal{M}$ are represented by finite dimensional vector spaces $U = \mathbb{R}^{d_U}, V = \mathbb{R}^{d_V}, M = \mathbb{R}^{d_M}$, which come from discretization schema. One can define corresponding finite dimensional strong and weak forms R, r, q that correspond to the respective operators and forms in the finite dimensional context.

In order to resolve pertinent physics or model structure in the state equation high dimensional discretization schema are often required. For example, this is the case for global scale models for earth physics, such as climate models or earth mantle physics. Since the state equation operator $R(\cdot, m)$ is nonlinear, solutions u are found via nonlinear iteration. This is the main computational bottleneck to evaluating the quantity of interest $q(u(m))$. Take for example the situation where $U = V'$ are Hilbert spaces such as in the continuous Galerkin finite element approximation of PDEs. Via nonlinear iteration one finds a solution $u \in U$ by positing an initial guess u_0 and updating the guess via Newton's method:

$$\partial_u R(u_i, m)\delta u = -R(u_i, m) \tag{2.1.6a}$$

$$u_i = u_i + \delta u. \tag{2.1.6b}$$

The iteration terminates when a suitable error tolerance $\|R(u_i, m)\|_U \leq \epsilon$ is achieved. At each iteration, formation of the Jacobian $\partial_u R(u_i, m)$ requires $O(d_U^2)$ work, and direct inversion formally requires $O(d_U^3)$ work. This is untenable for large discretization dimensions, so iterative Krylov solvers are typically used. Krylov methods do not require formation of the entire matrix, and instead iteratively build an approximation to the linear solution by using matrix vector products with the operator. This reduces the computational cost to $O(kd_U)$. The prefactor k however will depend on the spectral properties and conditioning of the Jacobian. Domain specific information may

be required to develop preconditioning methods to make the computational cost of approximate inversion tenable.

Due to the computational difficulty in evaluating the mapping $m \mapsto u$, direct evaluation of the mapping $m \mapsto u \mapsto q$ is out of the question for many-query / outer loop applications such as statistical inference, optimization etc. In what follows I consider q as a function of m, $q(m) = q(u(m))$, so when I speak of derivatives of q I mean the implicit derivatives of q with respect to m: $\nabla q = \nabla_m q(m)$.

2.1.2 Input Projection: Finding model structure with first derivatives

In the setting of parametric mappings, the input parameter m comes from a discretization of an infinite dimensional parameter. Algorithmic complexity suffers from the curse of dimensionality as the discretization dimension grows. In order to avoid the curse of dimensionality I seek to exploit low-dimensional structure of the model in architecting neural network surrogates.

One typical approach to exploit low dimensional structure of the input space is to use Karhunen-Loève decompositions [120], which exploits low dimensional correlation structure of ν. This approach does not take into account how the output response is informed by the input space. Gradient based input reduction methods can be used to find a global subspace for which the output response is sensitive to the input, as characterized by the first derivative of the output response with respect to the input. Similar techniques have been popularized for dimension reduction for scalar valued functions under the name "active subspaces" [37], bounds for optimal error bounds can be established using Poincaré inequalities. The ideas have been generalized to vector valued functions [139], or scenarios where Poincaré inequalities do not hold [102].

I start by reviewing some of the theory as it pertains to this approach, see [139] for more detailed exposition. The goal is to use a low rank approx-

imation for the ν averaged Gauss-Newton Hessian:

$$\mathbb{E}_\nu[\nabla q^T \nabla q] = \int_M \nabla q(m)^T \nabla q(m) d\nu(m) \in \mathbb{R}^{d_M \times d_M} \tag{2.1.7}$$

as a reduced basis for the input space $\mathbb{R}^{d_M}$. The orthonormal basis from the low rank eigenvalue decomposition for this quantity represents directions for which the response function q is globally sensitive to the input parameter $m \in \mathbb{R}^{d_M}$.

An effective strategy for constructing an input dimension reduced surrogate is to construct an optimal ridge function approximation of $m \mapsto q$. A ridge function is a composition of the form $g \circ h$, where $h : \mathbb{R}^{d_M} \to \mathbb{R}^r$ is a linear mapping (a tall skinny matrix), and $g : \mathbb{R}^r \to \mathbb{R}^{d_Q}$ is a measurable function. Input dimension reduced dense neural networks generally have this form.

For the sake of analysis, one can analyze the low dimensional representation as being embedded in the higher dimensional space. Consider the ridge function parametrizations given by the $g(P_r m)$, where $P_r \in \mathbb{R}^{d_M \times d_M}$ is a rank-r projector. Given a tolerance $\epsilon > 0$ the problem is to find a projector P_r and a ridge function g such that

$$\|q - g \circ P_r\|_{\mathcal{H}} \leq \epsilon. \tag{2.1.8}$$

In [139] optimal ridge functions over the Hilbert space $\mathcal{H} = L^2(\mathbb{R}^{d_M}, \nu; \mathbb{R}^{d_Q})$ are investigated. The probability ν is taken to be a Gaussian: $\nu = \mathcal{N}(m_0, \Sigma)$. They establish a bound for approximation of q by a conditional expectation of q with respect to the subspace defined by the projector P_r, that is based on the trailing eigenvalues of the prior preconditioned Gauss-Newton Hessian of the scalar function $\|q(m)\|^2_{\ell^2}$. In the case that the projector P_r is orthgonal with respect to the prior, the conditional expectation with respect to the sigma-algebra generated by P_r can be written as

$$\mathbb{E}_\nu[q|\sigma(P_r)](m) = \mathbb{E}_{y\sim\nu}[q(P_r m + (I_{d_M} - P_r)y)], \tag{2.1.9}$$

this integral marginalizes out the orthogonal complement of the span of P_r. The Gauss-Newton Hessian of $\frac{1}{2}\|q(m)\|^2_{\ell^2(\mathbb{R}^{d_Q})}$ is

$$H_{\mathrm{GN}} = \int_{\mathbb{R}^{d_M}} \nabla q(m)^T \nabla q(m) d\nu(m). \tag{2.1.10}$$

If (λ_i, v_i) are the generalized eigenpair for the system

$$H_{\mathrm{GN}} v_i = \lambda_i \Sigma^{-1} v_i, \tag{2.1.11}$$

then a bound for the approximation error $\|q - \mathbb{E}_\nu[q|\sigma(P_r)]\|^2_{\mathcal{H}}$ can be obtained by the Poincaré inequality:

$$\|q - \mathbb{E}_\nu[q|\sigma(P_r)]\|^2_{\mathcal{H}} \leq \sum_{i=r+1}^{d_M} \lambda_i^M, \tag{2.1.12}$$

when the projector is taken to be $P_r = V_r V_r^T \Sigma^{-1}$. See Proposition 2.6 in [139] for more information. This bound establishes that when the spectrum of the Gauss-Newton Hessian decays fast, the mapping $m \mapsto q$ can be well approximated in expected value by a ridge function that is restricted to only the informed modes of the input space.

2.1.3 Output Projection: Proper Orthogonal Decomposition

Due to practical constraints the output for the mapping $m \mapsto q$ will always be a finite dimensional vector space, however in principle $\mathbb{R}^{d_Q}$ could represent a discretized approximation of an infinite dimensional field. In either case, exploiting low dimensional structure for the output is of practical interest in addition to the problem of finding low dimensional representation of input space.

Reduced order modeling (ROM), and particularly the reduced basis method (RBM) have been developed to help reduce the dimension of output space for PDE mappings [12, 29, 30, 107, 132]. This methodological framework has made tractable the solution of many-query problems involving PDEs (optimization, inference, control, inverse problems etc.) [18]. In these methods low dimensional representations of output space are made via snapshots of

the output space from data. This is related to principal component analysis (PCA) done in machine learning applications.

The proper orthogonal decomposition (POD) for the output space is the eigenvalue decomposition of the expectation of the following matrix, which is nonsingular when $q \in \mathcal{H} = L^2(\mathbb{R}^{d_M}, \nu; \mathbb{R}^{d_Q})$:

$$\mathbb{E}_\nu[qq^T] = \int_{\mathbb{R}^{d_M}} q(m)q(m)^T d\nu(m) = \Phi D \Phi^T. \tag{2.1.13}$$

By results related to the Hilbert-Schmidt Theorem, the r_Q rank POD is a minimizer of the following minimization problem

$$\begin{aligned} \min_{W \in \mathbb{R}^{d_Q \times r_Q}} \int_{\mathbb{R}^{d_M}} \|q(m) - WW^T q(m)\|^2_{\ell^2(\mathbb{R}^{d_Q})} d\nu(m) \\ W^T W = I_{r_Q}. \end{aligned} \tag{2.1.14}$$

See for example [82, 107], I restate this result here.

Proposition 2.1 (Proposition 2.1 in [82])**.** *The POD basis $\Phi \in \mathbb{R}^{d_Q \times r_Q}$ is such that*

$$\begin{aligned} &\int_{\mathbb{R}^{d_M}} \|q(m) - \Phi\Phi^T q(m)\|^2_{\ell^2(\mathbb{R}^{d_Q})} d\nu(m) \\ = &\min_{W \in \mathbb{R}^{d_Q \times r_Q}, W^T W = I_{r_Q}} \int_{\mathbb{R}^{d_M}} \|q(m) - WW^T q(m)\|^2_{\ell^2(\mathbb{R}^{d_Q})} d\nu(m). \end{aligned} \tag{2.1.15}$$

And the error is given by the trailing eigenvalues of K:

$$\int_{\mathbb{R}^{d_M}} \|q(m) - \Phi\Phi^T q(m)\|^2_{\ell^2(\mathbb{R}^{d_Q})} d\nu(m) = \sum_{i=r_Q+1}^{rank(T)} \lambda_i^Q \tag{2.1.16}$$

In typical applications, one does not have the ability to explicitly evaluate the integral with respect to ν, Instead one can draw samples $m_i \sim \nu$ for which one can evaluate the response $q_i = q(m_i)$ at finitely many points. This leads to the Monte Carlo approximation from N_{data} samples:

$$\mathbb{E}_\nu[qq^T] \approx \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} q_i q_i^T \in \mathbb{R}^{d_Q \times d_Q} = \Phi D \Phi^T. \tag{2.1.17}$$

A calculation of a low rank approximation of $\Phi D \Phi^T$ from samples is referred to as the method of snapshots since the training data q_i are taken to be "snapshots" of the output. The approximation is be bounded in expectation by the typical Monte Carlo bound (square root of upper bound for the trace covariance of $[qq^T]_{ij}$ divided by $\sqrt{N_{\text{data}}}$).

Like active subspace, POD serves as a constructive prescription for a low rank basis that is optimal in sense. It also serves as a means of reliably calculating the inherent dimensionality of the output space for the mapping $m \mapsto q$.

2.1.4 Input-Output Error Bound for Optimal Ridge Function

Parametric mappings for which the eigenvalues of both the active subspace and POD operators decay rapidly can be well approximated by restriction to these input and output subspaces. This is a specific type of ridge function, where the input data are projected down to their representation in the dominant modes of the active subspace decomposition, the nonlinear ridge function represents the mapping from the dominant modes of the input to the dominant modes of the output which then represent the output on the POD basis.

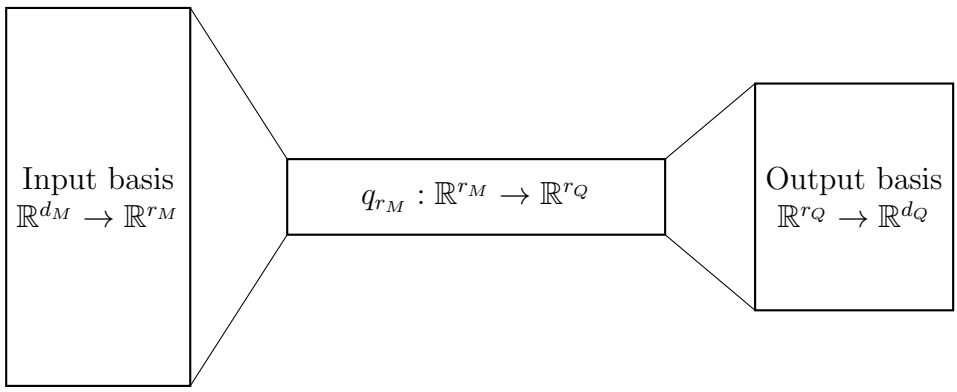


Figure 2.2: Dimension reduced representation by conditional expectation ridge function

Combining the active subspace and POD approach, an error bound can be established for the conditional expectation ridge function that is restricted

to the dominant modes of the POD basis.

Proposition 2.2 (Input-Output Ridge Function Error Bound)**.** *Let* $P_{r_M} = V_r V_R^T \Sigma^{-1} \in \mathbb{R}^{d_M}$ *be the projectors coming from the active subspace generalized eigenvalue problem* (2.1.11)*, and define the conditional expectation of the output* q *with respect to the sigma algebra generated by this projector:*

$$q_{r_M}(m) = \mathbb{E}_\nu[q|\sigma(P_{r_M})](m). \tag{2.1.18}$$

Define the rank r_Q *POD decomposition for* q_{r_M} *as follows:*

$$\left[\mathbb{E}_\nu[q_{r_M} q_{r_M}^T]\right]_{r_Q} = \left[\int_{\mathbb{R}^{d_M}} q_{r_M}(m) q_{r_M}(m)^T d\nu(m)\right]_{r_Q} = \widehat{\Phi}_{r_Q} \widehat{D}_{r_Q} \widehat{\Phi}_{r_Q}^T. \tag{2.1.19}$$

Then one can obtain the following bound:

$$\int_{\mathbb{R}^{d_M}} \|q(m) - \widehat{\Phi}_{r_Q} \widehat{\Phi}_{r_Q}^T q_{r_M}(m)\|_{\ell^2(\mathbb{R}^{d_Q})}^2 d\nu(m) \leq \sum_{i=r_M+1}^{d_M} \lambda_i^M + \sum_{i=r_Q+1}^{d_Q} \lambda_i^Q \tag{2.1.20}$$

Proof. This result follows from the triangle inequality

$$\begin{aligned} &\int_{\mathbb{R}^{d_M}} \|q(m) - \widehat{\Phi}_{r_Q} \widehat{\Phi}_{r_Q}^T q_{r_M}(m)\|_{\ell^2(\mathbb{R}^{d_Q})}^2 d\nu(m) \\ \leq &\int_{\mathbb{R}^{d_M}} \|q(m) - q_{r_M}(m)\|_{\ell^2(\mathbb{R}^{d_Q})}^2 + \|q_{r_M}(m) - \widehat{\Phi}_{r_Q} \widehat{\Phi}_{r_Q}^T q_{r_M}(m)\|_{\ell^2(\mathbb{R}^{d_Q})}^2 d\nu(m), \end{aligned} \tag{2.1.21}$$

and application of (2.1.12) and Proposition 2.1. □

This result establishes that when the spectra for $\mathbb{E}_\nu[\nabla q^T \nabla q]$ and $\mathbb{E}_\nu[q q^T]$ decay quickly, low dimensional function approximation with ridge functions can achieve high accuracy in expectation with respect to ν. The contributions of the orthogonal complements of the dominant bases of active subspace and POD make minimal contributions to informing the input-output map when the eigenvalue decay for these decompositions is fast. This error bound motivates the approach for an input-output dimension reduced neural network.

2.2 Using Model Structure to Aid Architecture Design

The low rank approximations of the subspaces for active subspace and POD expose both the intrinsic dimensionalities of the input and output spaces, as well as orthonormal bases for representing the dominant linear subspaces of the input and output spaces, that are have error bounds characterized in the preceding sections.

In this section I discuss strategies for constructing neural network architectures using these building blocks. A dimension independent neural network can be formulated as follows:

$$f(m, w) = \Phi_{r_Q} f_r(V_{r_M}^T m, w). \tag{2.2.1}$$

Where f_r denotes the dense neural network in the low dimensional representation. In this case the size of the configuration space scales with r_M and r_Q (the input and output inherent dimensions), not the larger dimensions d_M, d_Q. The dependence on the input and output dimensions are removed via the use of the projections Φ_{r_Q} and V_{r_Q}.

The question is then how to parametrize the neural network in the low dimensional representation. A first naïve idea would be to use pure dense blocks, but these networks can be difficult to train, and the configuration space can quickly grow quickly as $O(N_{\text{layers}} r_M r_Q)$ can become somewhat large. One idea to avoid this configuration space blow-up is based on convolutional residual networks (resnets). Convolutional resnets are a popular architecture in machine learning applications [61]. The basic building block for each layer is a nonlinear compact perturbation of identity:

$$x_{i+1} = x_i + C_{i1}\sigma_i(C_{i0}x_i), \tag{2.2.2}$$

where C_{i0}, C_{i1} are the convolution and deconvolution operations, and the function σ_i is a nonlinear activation function at layer i. Convolution operators compress information by adding up nearby values in an array that are weighted by a convolutional kernel operator; this operation is a discretization

of the related integral convolution operator. The entire convolution operation can be represented as matrix products, where the convolution matrix is a cyclic (Toeplitz-like) matrix function of the convolutional kernel. Compact perturbations of identity have nice properties, especially if the nonlinear activation function is smooth; they can be adapted to be invertible and diffeomorphic if restrictions are imposed on the Jacobians of each layer [73].

Convolution operations are sparse so the dimension of the weights do not grow very fast, and can be made fast to evaluate on modern GPU computer architectures. Convolutions are useful for compressing 2D and 3D image data, but are not as useful for models with non-spatial information. Further, the kernels are learned during neural network training, and it is not clear how to use a priori model information to infer the dimensionality of the kernels, or infer good initial guesses for the kernels, as is the case in the approach outlined here using model based projectors. The sparsity pattern imposed by convolution operations has nice properties, and motivates the approach I use for the parametrization of the low dimensional neural network representation. I impose a sparsity pattern using low rank matrix algebra to build a sparsified residual network layer:

$$x_{i+1} = x_i + W_{i1}\sigma_i(W_{i0}x_i). \tag{2.2.3}$$

The matrices $W_{i1} \in \mathbb{R}^{r_M \times r_i}$ and $W_{i0} \in \mathbb{R}^{r_i \times r_M}$ are low rank if $r_i \ll r_M$, so each layer used in the internal neural network contributes only a few model parameters. If the residual network building blocks are to be used for each layer in the inner network, then one must take $r_M = r_Q$, or use a different rectangular layer to make the dimensions match at the beginning or end of the residual block chain of layers. For a sequence of fixed rank compact perturbations of identity the size of the configuration space is $O(N_{\text{layers}} r_M k_i) = O(N_{\text{layers}} r_Q k_i)$. This neural network construction is parsimoniously constructed in that global projectors that expose the inherent dimensionality of the mapping are used to compress the input and output

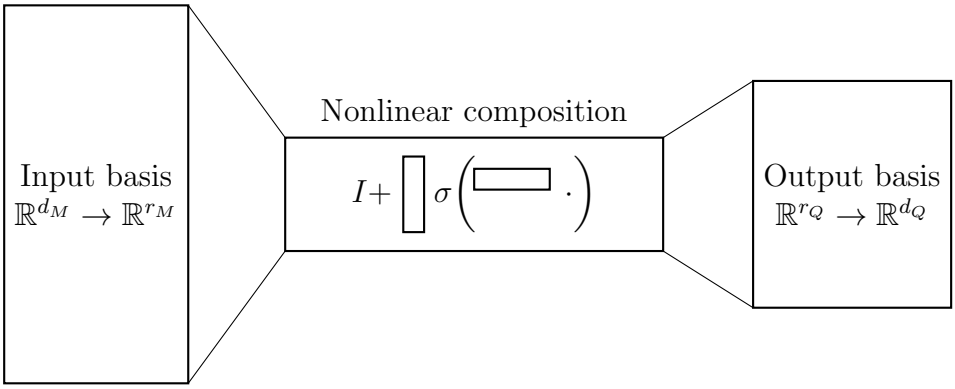


Figure 2.3: Projected low rank residual network (pLRRN)

spaces. Further, the inner neural network parametrization only scales linearly with the inherent dimensionality (as opposed to quadratically for a naïve dense parametrization of each layer). I term this network architecture projected low rank residual neural networks (pLRRN) and this is the main type of architecture I study due to its compact formulation. A schematic for this general idea is shown below.

Training this model can be made very cheap since the configuration space is small. Since the configuration space is small, only a small amount of training data is necessary to infer the parameters in the model, which is an important feature for the setting of surrogates for expensive parametric maps. The model only attempts to represent the nonlinear mapping $m \mapsto q$ in dominant modes of the input and output space, this can be thought of as a sort of regularization, where highly oscillatory modes of the input-output mapping that do not inform the mapping are suppressed by construction. This is similar to the use of Tikhonov regularization and the Morozov discrepancy principle in ill-posed optimization problems [128, 118]. The neural network training problem is itself an ill-posed inverse problem; its ill-posedness is studied in Appendix 1.

The weights for the neural network $\{[W_{i1}, W_{0i}]\}_{i=1}^{N_{\text{layers}}}$ are found via solution of a stochastic nonconvex optimization problem. One can also allow that the input and output projectors be trained, however this will increase

the size of the configuration space to $O(d_M r_M + N_{\text{layers}} r_Q k_i + r_Q d_Q)$; the configuration space explicitly depends on both the input and output dimensions. One is however likely to improve on the solution at least marginally; if d_M and d_Q are not too big than this can be desirable. If d_M or d_Q are very large, the marginally improved performance may not be amortized by a costly high dimensional training problem.

The neural network training problem is NP-hard, and in practice one must settle for local minima. The quality of the trained neural network is heavily dependent on initial guess. In the case that the first and last layer are also to be trained, the projectors that come from the active subspace and POD decompositions could be thought of a very good initial guesses, since they restrict the mapping to informed modes of the input and output. In this case, first training the projected neural network before freeing up the first and last dense layers would be pragmatic.

2.3 Scalable Computation of the Input and Output Projectors

The matrices $\mathbb{E}_\nu[\nabla q^T \nabla q]$, $\mathbb{E}_\nu[qq^T]$ can be approximated in a scalable and efficient manner using sampling and matrix-free randomized linear algebra. The integrals can be approximated as finite sums via Monte Carlo approximation; given draws $m_i \sim \nu$, one can generate training data $\{(m_i, q_i)\}_{i=1}^{N_{\text{data}}}$, and approximate the POD basis as in the method of snapshots (2.1.17). The integral $\mathbb{E}_\nu[\nabla q^T \nabla q] \in \mathbb{R}^{d_M}$ can be approximated via the action of the matrices on Gaussian random vectors. The integral approximation is obtained via Monte Carlo, for $\omega_M \in \mathbb{R}^{d_M}$:

$$\mathbb{E}_\nu[\nabla q^T \nabla q]\omega_M \approx \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \nabla q(m_i)^T \nabla q(m_i)\omega_M \tag{2.3.1}$$

This matrix-free computation requires the of the action of operators $\nabla q(m_i) \in \mathbb{R}^{d_Q \times d_M}$ and $\nabla q(m_i)^T \in \mathbb{R}^{d_M \times d_Q}$ at various points $m_i \in \mathbb{R}^{d_M}$. The Monte Carlo sum leverage modern computing architecture and use sample parallelism, making the computation for many samples tractable in terms of time

and memory.

2.3.1 The action of ∇q and ∇q^T using adjoints

The implicit derivative of the output function $q(m) = q(u(m))$ with respect to the the input parameter m can be found via a Lagrange multiplier approach to incorporate the implicit dependence on the weak form of the state equation. Since $q(m) \in \mathbb{R}^{d_Q}$, the result is derived for each component:

$$\mathcal{L}_i(u, m, v) = q(u(m))_i + v_i^T R(u, m), \tag{2.3.2}$$

or equivalently take all derivatives at once, with the adjoint variable matrix $V \in \mathbb{R}^{d_V \times d_Q}$:

$$\mathcal{L}(u, m, v) = q(u(m)) + V^T R(u, m). \tag{2.3.3}$$

A variation of $\mathcal{L}$ with respect to the adjoint variable matrix yields the state equation:

$$R(u, m) = 0, \tag{2.3.4}$$

which is solved for u, given a draw m. A variation of $\mathcal{L}$ with respect to the state variable u yields the adjoint equation:

$$\underbrace{\frac{\partial q}{\partial u}}_{B} + V^T \underbrace{\frac{\partial R}{\partial u}}_{A} = 0 \tag{2.3.5}$$

which is solved for each adjoint variable:

$$V^T = -BA^{-1}. \tag{2.3.6}$$

Lastly a variation of $\mathcal{L}$ with respect to the input parameter m allows for the formation of the Jacobian:

$$\nabla_m q + V^T \underbrace{\frac{\partial R}{\partial m}}_{C} = 0. \tag{2.3.7}$$

So finally one can compute

$$\nabla_m q(u(m)) = BA^{-1}C = \frac{\partial q}{\partial u}\left[\frac{\partial R}{\partial u}\right]^{-1}\frac{\partial R}{\partial m}. \tag{2.3.8}$$

To evaluate at a point m_i one must solve the state and adjoint equations first before applying the operator in a direction $\widehat{m} \in \mathbb{R}^{d_M}$:

$$\nabla_m q(u(m_i))\widehat{m} = BA^{-1}C\widehat{m}. \tag{2.3.9}$$

The state and adjoint equations only need to be calculated once before the action of the operator in many directions $\widehat{m}$ is formed. Similary the transpose of the operator can be applied to $q \in \mathbb{R}^{d_Q}$ via

$$\nabla_m q(u(m))^T = C^T A^{-T} B^T \widehat{q}. \tag{2.3.10}$$

The key observation about the action of ∇q and ∇q^T is that unlike the forward solution of the PDE state equation they do not involve any nonlinear iterations. They only involve linearizations of the PDE. These matrices can be efficiently computed and applied over and over to many right hand sides, which makes the cost of evaluating the derivatives via adjoint methods cheaper than the forward solution of the model. The expensive part of the derivative evaluation is the solution of the forward nonlinear PDE, which is required for training data generation anyways. Thus one can save computation by generating training data and approximations of derivative subspaces in tandem. The computational cost of computing derivative subspaces of the output response is marginally less expensive than the evaluation of the nonlinear mapping.

2.3.2 Approximation with randomized linear algebra

The single pass randomized eigenvalue decomposition algorithms can approximate the k low rank eigenvalue decomposition of a matrix $Q_k D_k Q_k^T \approx A \in \mathbb{R}^{n \times n}$ for $k + p$ matrix vector products, with expected approximation error bounded by

$$\mathbb{E}_\rho[\|A - Q_k D_k Q_k^T\|] \leq \left(1 + 4\frac{\sqrt{n(k+p)}}{p-1}\right)|d_{k+1}|. \tag{2.3.11}$$

See [58, 89], More accuracy can be obtained via double pass or power iteration, but this requires more applications of the operators. Here $\mathbb{E}_\rho$ denotes

the expectation taken with respect the the Gaussian measure ρ from which the random matrix $\Omega \in \mathbb{R}^{n+(k+p)}$ is sampled. Higher accuracy in the approximation can be achieved for more applications of the operator (i.e. power iteration, multi-pass methods). Automated procedures such as adaptive range finding can be used to find the rank k such that a specific tolerance is met, i.e. for a given $\epsilon > 0$ find k, p such that

$$\left(1 + 4\frac{\sqrt{n(k+p)}}{p-1}\right)|d_{k+1}| \leq \epsilon. \tag{2.3.12}$$

The matrix approximation is formed via the action of the operator on Gaussian random vectors which is guaranteed to resolve dominant modes of the operator due to concentration of measure, matrix concentration inequalities such as Bernstein or Chernov. This is an inherently scalable process since all of the matrix vector products are independent of one another, i.e. they can be efficiently parallelized and high levels of concurrency can be leveraged. This is in contrast to other matrix-free approximations such as Krylov, where the matrix vector products are inherently serial.

2.4 Numerical Experiments

In this section, I evaluate the performance of model based projected neural network architectures on some PDE inference problems. Once the architectures are decided (as discussed in the preceeding section), the optimal weights $w^* \in \mathbb{R}^{d_W}$ are found via solving an empirical risk minimization problem, in this case least squares. Given training data $X_{\text{train}} = \{(m_i, q(m_i))\}_{i=1}^{N_{\text{train}}}$, candidate optimal weights are found via solving the regularized empirical risk minimization problem,

$$\min_{w \in \mathbb{R}^{d_W}} F_{X_{\text{test}}}(w) = \frac{1}{2N_{\text{test}}}\|q(m_i) - f(m_i, w)\|^2_{\ell^2(\mathbb{R}^{d_Q})} + \frac{1}{2}\|w\|^2_R, \tag{2.4.1}$$

which can be viewed as a Monte Carlo approximation of the regularized expected risk minimization problem over the true measure ν:

$$\min_{w \in \mathbb{R}^{d_W}} F_{X_{\text{test}}}(w) = \frac{1}{2}\int_{\mathbb{R}^{d_M}} \|q(m_i) - f(m_i, w)\|^2_{\ell^2(\mathbb{R}^{d_Q})} d\nu(m) + \frac{1}{2}\|w\|^2_R \tag{2.4.2}$$

The positive definite matrix $R \in \mathbb{R}^{d_W \times d_W}$ defines a regularization for the problem. In this case I take $R = \gamma I_{d_W}$, i.e. Tikhonov regularization [128], for some paramter $\gamma > 0$. This parameter can be found via Morozov discrepancy or other methods [118]. I used a subsampled inexact Newton CG method because it performed reliably better than other methods on this problem [96, 97] or Chapter 3.

The various architectures are evaluated on their average performance on unseen data. I use various meta-parameters to evaluate the performance of the architectures.

1. The cardinality of the training data corpus N_{train}. I am specifically interested in methods that can generalize well for small corpi of training data.

2. Since the empirical risk minimization problem (2.4.2) is nonconvex, and one has to settle for local minima that are heavily dependent on both initial guess as well as data stochasticity, I report sample average generalization errors taken over various initial guesses and data subsets.

If the trained neural network model is $f_w(m)$, I define the relative error as

$$\text{Relative Error} = \sqrt{\frac{\sum_{i=1}^{N_{\text{test}}} \|q_i - f_w(m_i)\|^2}{\sum_{i=1}^{N_{\text{test}}} \|q_i\|^2}}, \tag{2.4.3}$$

and the accuracy as

$$\text{Accuracy} = 100(1 - \text{Relative Error}). \tag{2.4.4}$$

Note that for very poor approximations, accuracy can be negative. For all problems I test against a linear Taylor approximation of the map $m \mapsto q$ of the form:

$$q_{\text{lin}}(m) = q(m_0) + \nabla q(m_0)^T (m - m_0). \tag{2.4.5}$$

Taylor approximations are very useful local approximations of parametric maps [33, 34], and represent a baseline benchmark to compare neural network surrogates against. The performance of the linear Taylor approximation

also gives a general idea of how difficult the problem is: when it performs well the map is approximately linear, when it performs poorly nonlinearity and variance of the output make the mapping difficult to approximate for a surrogate model. For each neural network model I use a last layer initial guess of $q(m_0)$, i.e. the quantity of interest evaluated at the mean. This idea is motivated by the idea of a Taylor approximation or a model that has a mean component plus a nonlinear perturbation. The neural networks all performed better when this initial guess was used.

I investigate two numerical examples which involve parameter to observable maps related to PDE inference problems.

The first example is a 3D nonlinear ice dynamics inference problem, where the model parameter m represents an unknown boundary condition for an ice flow model. The output quantity of interest in this example are observations of the velocity and pressure for the ice sheet at points located on the surface of an ice sheet.

The second example is a 2D quasi-linear convection-reaction-diffusion PDE, where the model parameter m represents an uncertain volumetric forcing term, and the output quantity of interest is the PDE state (concentration) at some points on the interior of the PDE domain.

In both examples I demonstrate that the projected neural network models can perform as well and sometimes better than the conventional approaches at reconstructing the nonlinear parametric map, when the discretization dimension is not too large. For the second numerical example I demonstrate that as the discretization dimension grows, the conventional data-driven approach performs worse and worse. In this case the projected approach performs comparable, or sometimes even better as the data dimension grows, while the configuration dimension for the neural network stays fixed for each problem.

Both problems use a Gaussian prior for the uncertain parameter m, with Matern covariance operator that involves fractional PDE operators.

2.4.1 Model for uncertain parameter m

The following examples are PDEs in 2D and 3D that involve an uncertain model parameter. In the 2D convection diffusion problem the uncertain parameter represents a coefficient for a nonlinear reaction term defined in the domain $\Omega_M \subset \mathbb{R}^2$. In the 3D ice dynamics problem, the uncertain parameter represents a Robin boundary condition defined on the bottom of the 3D domain : $\Omega_M \subset \partial\Omega \subset \mathbb{R}^3$. In both cases the model parameter is sampled from a Gaussian probability distribution with a Matern covariance prior.

$$C = (\delta I - \gamma \nabla \cdot (\Theta \nabla))^{-\alpha} \tag{2.4.6}$$

The uncertain parameter m is the solution of a linear fractional stochastic PDE [80]. The choice of $\alpha > d/2$ where d is the spatial dimension of the PDE, makes the covariance trace class, in both problems I take $\alpha = 2$. When the covariance is trace class this guarantees that the uncertain parameter m is L^2 integrable, which makes the PDE well-posed. The parameters $\delta, \gamma > 0$ control the correlation length as well as the marginal variance of the probability distribution. The matrix $\Theta \in \mathbb{R}^{d \times d}$ introduces spatial anisotropy [34]. When the PDE problem is discretized the vector representation of m has dimension d_M and I consider m to be a vector in $\mathbb{R}^{d_M}$.

The correlation length for draws from the prior is controlled by the ratio δ/γ, and for a fixed correlation length, larger values of γ and δ reduce the marginal variance for distribution.

2.4.2 Ice Sheet Model

The first parametric mapping comes from a nonlinear Stokes ice sheet model with uncertain basal sliding field parameter. Let $\Omega \subset \mathbb{R}^3$ refer to the ice sheet domain, and define the bottom boundary of the ice sheet as Γ_{bot}. For this inference problem, the uncertain parameter m is an uncertain Robin boundary condition on Γ_{bot} that varies from a no slip boundary condition to a free slip boundary condition. The uncertainty in this parameter is due to the inability to observe the physics of the ice sheet at its base. This parameter

has to be inferred via sparse observations of the top of the ice sheet and the PDE model [65]. The state variable, $u = (v, p) \in [H^1(\Omega)]^3 \otimes L^2(\Omega)$ is a velocity and pressure pair defined throughout the domain of the PDE. The nonlinear Stokes problem with uncertain basal sliding parameter is defined below.

$$-\nabla \cdot \sigma = \rho g \text{ in } \Omega \tag{2.4.7a}$$

$$\nabla \cdot v = 0 \text{ in } \Omega \tag{2.4.7b}$$

$$\sigma = \mu(v)D(v) - Ip \tag{2.4.7c}$$

$$D(v) = \frac{1}{2}[\nabla v + \nabla v^T] \tag{2.4.7d}$$

$$\mu(v) = \mu_0[D(v) : D(v)]^{-\frac{1}{3}} \tag{2.4.7e}$$

$$\sigma \cdot n = 0 \text{ on } \partial\Omega \setminus \Gamma_{\text{bot}} \tag{2.4.7f}$$

$$v \cdot n = 0 \text{ on } \Gamma_{\text{bot}} \tag{2.4.7g}$$

$$\tau(e^m v + \sigma \cdot n) = 0 \text{ on } \Gamma_{\text{bot}} \tag{2.4.7h}$$

$$\tau = (I - nn^T) \tag{2.4.7i}$$

$$q(m) = Bu(m) = [u(x_i, m)] \quad \text{at } x_i \in \Gamma_{\text{top}}. \tag{2.4.7j}$$

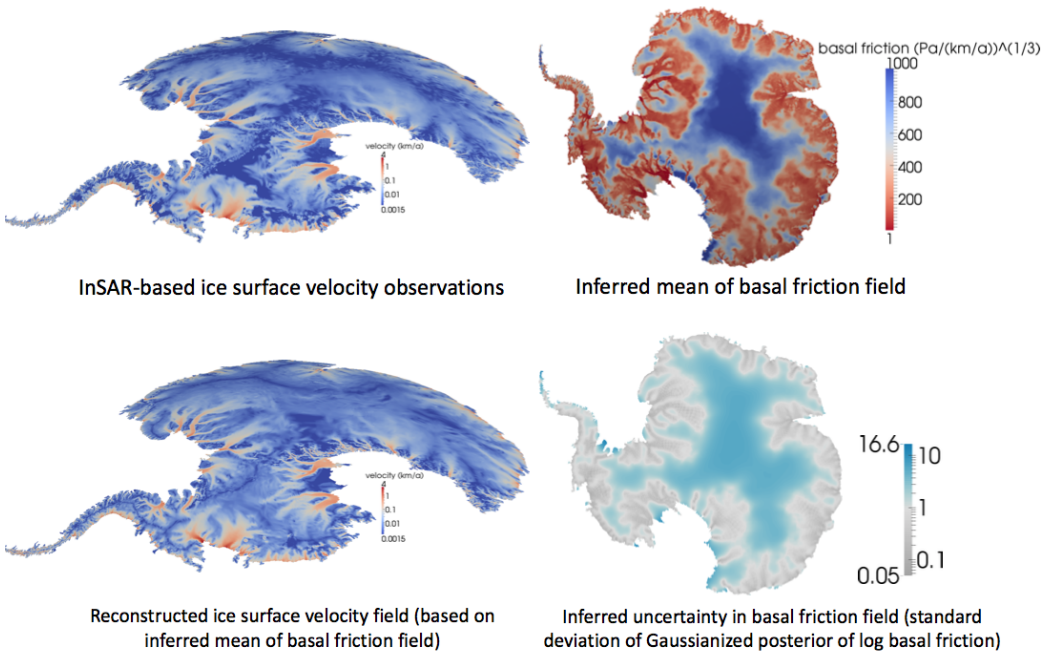
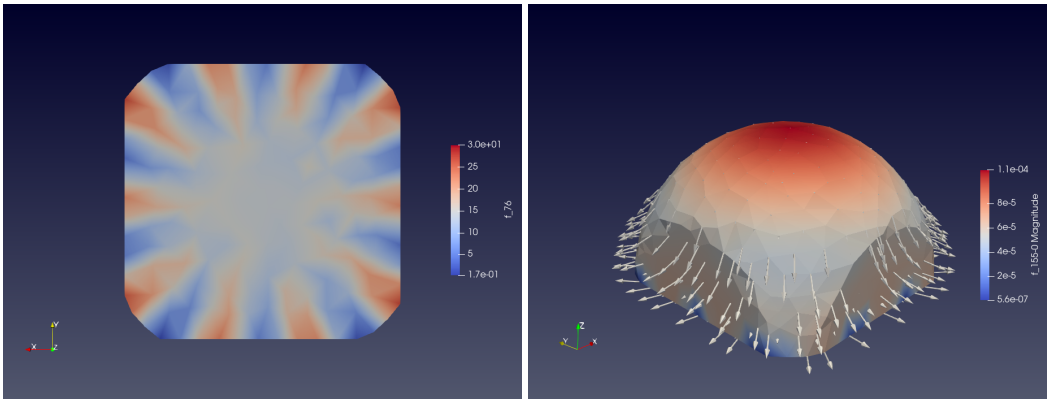


Figure 2.4: Antartic ice sheet with basal sliding field parameter [65]

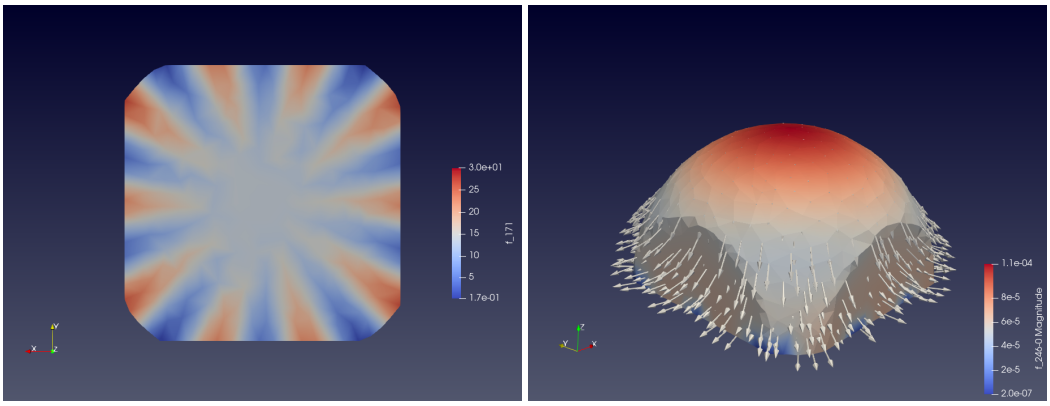
In this problem the quantity of interest is observations of the state $\approxeq = (v, p)$ at N_{targets} near the top of the ice sheet. This observable was chosen for its relevance to inverse problems relating to polar ice sheets, where the surface velocity is observed and one wants to invert for the basal sliding field [65]. I sample $N_{\text{targets}} = 25$ points near the top surface of the dome of the ice sheet. The dimension of q is thus 100.

Two different meshes levels of refinement of the same mesh were used. For the first mesh the parameter dimension is $d_M = 1,015$, and for the second the parameter dimension is $d_M = 2,339$. I study many parametrizations of the Matern covariance prior. I take choices of $\gamma, \delta \in \{0.1, 0.25, 0.5, 1.0\}$. This allows for the study of problems with varying degrees of difficulty. Samples for the choice $\gamma = \delta = 1.0$ are shown below.



(a) Mean basal sliding field for coarse mesh for $\gamma = \delta = 1.0$

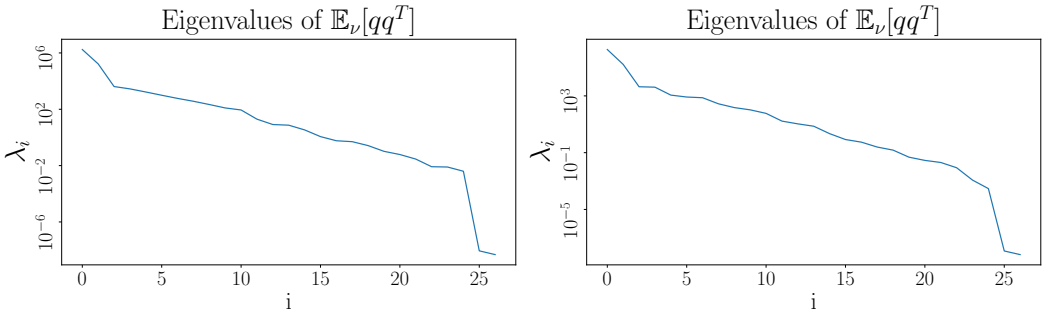
(b) Velocity at mean parameter for coarse mesh for $\gamma = \delta = 1.0$



(c) Sample basal sliding field for fine mesh for $\gamma = \delta = 1.0$

(d) Velocity at sample parameter for fine mesh for $\gamma = \delta = 1.0$

Mesh independence of the spectral decay can be seen for the following two plots of the POD spectrum can be seen for $\gamma = \delta = 1.0$.



(a) $d_M = 1,015$

(b) $d_M = 2,339$

Figure 2.6: POD eigenvalue decay for two different meshes, $\gamma = \delta = 0.1$

The active subspace eigenvalue decay for this problem agrees with the POD eigenvalue decay.

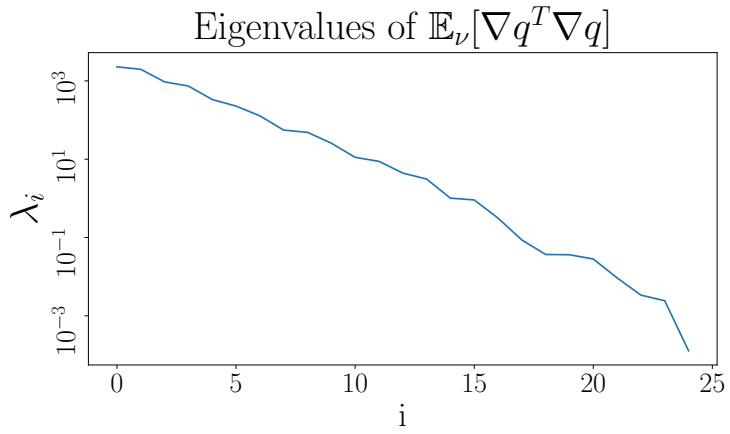


Figure 2.7: Active subspace eigenvalue decay for the finer mesh, $d_M = 2,339$

I investigate the pLLRN vs a discretization dimension dependent approach. I compare fixed ranks $r = 20, 40, 80$, this time with three nonlinear layers, against dense neural network parametrized by the output dimension $d_Q = 100$. The ranks were chosen to capture most of the dominant information $r = 20$ (the rank of the problem is somewhere around $20 - 25$ in general), and two oversampled ranks $r = 40, 80$ to see how the parsimonious projection parametrization effects the approximation. The inner ranks for each layer in the pLRRN are chosen to be 16 for this problem.

The dense neural network is parametrized by the dimensionality of the data, i.e. d_M and d_Q. The first layer is a dense block that maps $\mathbb{R}^{d_M} \to \mathbb{R}^{d_Q}$, and all other layers in the network are square maps from d_Q to d_Q. This allows for the most parsimonious fully dense neural network by mapping down to the output dimension right away. Note that the use of dense blocks that reduce the input dimension d_M to the smaller observation dimension d_Q will always have a configuration space dimension of at least $d_M d_Q$, so using fewer layers in this framework does not do much to reduce the configuration dimension. It is always the first layer that leads to the very large configuration dimension.

After some experimentation with activation functions I used softmax for the first few layers of the pLRRN, followed by softplus and identity for the last layer. Other activation functions were experimented with, such as sigmoid, tanh, as well as periodic functions which have shown some promise in PDE type problems [123], but the use of softmax for the first few nonlinear layers and then softplus for the last nonlinear layer performed the best experimentally. Initial guesses for the weights were sampled from Gaussian white noise $\mathcal{N}(0, I_{d_W})$. The configuration space dimension for the neural networks trained in the following examples are stated in the following table.

d_M	$1,015$	$2,339$
pLRRN $r = 20$	$2,760$	$2,760$
pLRRN $r = 40$	$5,420$	$5,420$
pLRRN $r = 80$	$10,740$	$10,740$
Dense 100	$152,100$	$284,500$

Table 2.1: Size of configuration spaces and parameter dimension for different meshes.

I am interested in parsimoniously parametrized neural network architectures that can be trained quickly. In the training problem I am curious which neural network architectures generalize to unseen data the best for a fixed number of accesses to the training data during training (i.e. neural network sweeps). A neural network sweep is defined as a forward and backward pass of the neural network. I allow the optimizer to access the entire data set 200 times. For a first order optimizer this would be equivalent to 200 epochs through the training data. However this is slightly different than epochs for a second order optimizer since a different set of data is used at each iteration for the Hessian vector products, which requires two forward and backward passed (sweeps) to form each Hessian vector product for each datum. The accuracy results are summarized below in the following two tables for the coarse mesh.

	$\gamma = 0.1$	$\gamma = 0.25$
$\delta = 0.1$	(-57., **42.2**, 40.2, 42.1, 40.1)	(-59., **40.2**, 37.5, 39.3, 37.6)
$\delta = 0.25$	(21.7, **63.9**, 63.5, 63.7, 63.4)	(20.9, **62.7**, 62.0, 62.0, 62.0)
$\delta = 0.5$	(54.5, **78.0**, **78.0**, 77.9, 77.9)	(56.0, **78.5**, 78.4, 78.4, 78.4)
$\delta = 1.0$	(72.5, **86.2**, 86.1, 86.1, 86.1)	(75.9, **88.1**, 88.0, 87.9, 87.9)

Table 2.2: Review of accuracy results for prior parameter sweep (Taylor linear, pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100), $d_M = 1,015$

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.1$	(-59., **39.6**, 37.0, 38.2, 36.9)	(-59., **39.5**, 36.5, 39.3, 36.3)
$\delta = 0.25$	(20.4, **62.3**, 61.8, 62.0, 61.7)	(20.2, 61.8, 61.6, **61.9**, 61.5)
$\delta = 0.5$	(56.2, **78.5**, **78.5**, 78.4, 78.4)	(56.3, **78.6**, 78.5, 78.4, 78.5)
$\delta = 1.0$	(76.9, **88.6**, **88.6**, **88.6**, 88.5)	(77.3, **88.9**, 88.7, 88.7, 88.8)

Table 2.3: Review of accuracy results for prior parameter sweep (Taylor linear, pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100), $d_M = 1,015$

For the first set of results all of the models perform roughly the same. The most parsimonious network, the pLRRN $r = 20$ obtained the best performance on all but one of the problems. In general the optimization problem was much easier for this problem, and all of the neural networks performed decently well, unlike some of the convection diffusion example where the optimizers could get stuck in bad regions of parameter space. These 16 problems suggest that a parsimoniously parametrized neural network is a good idea for the parametric map involving the effect of the basal sliding field on the surface velocity. The neural networks with more parameters tended to do worse for this problem.

For the finer mesh, I reduced the number of training data generated for the sake of computational time constraints. For this problem 200 training data were used, with all of it used for the gradient at each iteration, and 50 of those data sampled at each iteration for the Hessian vector products used in second order methods.

	$\gamma = 0.1$	$\gamma = 0.25$
$\delta = 0.1$	(-41., 40.7, 38.9, **40.8**, 37.8)	(-39., **44.0**, 43.8, 43.4, 43.2)
$\delta = 0.25$	(34.1, **68.9**, **68.9**, 68.8, 68.2)	(33.4, **68.3**, **68.3**, **68.3**, 67.3)
$\delta = 0.5$	(61.6, 80.8, **80.9**, 80.7, 80.1)	(63.2, 81.6, **81.7**, 81.5, 80.7)
$\delta = 1.0$	(76.3, **87.8**, **87.8**, 87.7, 86.9)	(80.1, **89.8**, **89.8**, 89.7, 88.8)

Table 2.4: Review of accuracy results for prior parameter sweep (Taylor linear, pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100), $d_M = 2,339$

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.1$	(-39., **44.3**, 43.1, 43.6, 42.8)	(-40., 44.8, **45.1**, 44.4, 42.2)
$\delta = 0.25$	(32.8, **68.1**, **68.1**, **68.1**, 67.5)	(32.4, **68.0**, **68.0**, **68.0**, 67.0)
$\delta = 0.5$	(63.1, **81.8**, **81.8**, 81.7, 80.6)	(62.9, **81.8**, 81.7, 81.5, 81.3)
$\delta = 1.0$	(80.8, 90.3, **90.4**, 89.9, 89.1)	(80.8, 90.3, **90.4**, 90.2, 89.1)

Table 2.5: Review of accuracy results for prior parameter sweep (Taylor linear, pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100), $d_M = 2,339$

In this set of examples one of the two most parsimonious neural networks obtained the best performance in every case. The parameter dimension about doubled from the previous mesh to this mesh, the pLRRN networks performed better in the case with larger parameter dimension. These results demonstrate that the data-driven approach performs only about as well as the projected model in the data informed subspaces of the input and output. As the discretization dimension grew the pLRRN began outperformed the data-driven approach by a slightly larger margin, but these results not conclusive.

The generation of training data for this model was very expensive and scaling the problem to very high parameter dimension was unfeasible. First the parameter lives in a 2D manifold within a 3D problem. The dimension for the nonlinear solve is the representation of the state variable in the 3D discretization. Further the 3D solver used a dense LU solver, and did not scale well. In order to make the problem scalable, an iterative method with a good preconditioner is important [65], this will be pursued in future work.

2.4.3 Convection Reaction Diffusion Problem

In order to show the key scaling benefits of the discretization dimension independent approach I investigate a 2D quasi-linear convection reaction diffusion problem with a nonlinear dependence on the random field m. The formulation of the problem is

$$-\nabla \cdot (k\nabla u) + \mathbf{v} \cdot \nabla u + u^3 = e^m f \quad \text{in } \Omega \tag{2.4.8a}$$

$$u = 0 \text{ on } \partial\Omega \tag{2.4.8b}$$

$$q(m) = Bu(m) = [u(\mathbf{x}^{(i)}, m)] \quad \text{at } \mathbf{x}^{(i)} \in \Omega \tag{2.4.8c}$$

$$\Omega = (0,1)^2. \tag{2.4.8d}$$

The observation points $\mathbf{x}^{(i)} \in \Omega$ are randomly sampled in the interior of the domain. In this problem the uncertain parameter shows up as a coefficient in volumetric forcing term for the reaction diffusion equation. The parameter has mean zero. The volumetric forcing function f is a Gaussian bump located at $x_1 = 0.7, x_2 = 0.7$.

$$f(x,y) = \max\left(0.5, e^{-25(x_1-0.7)^2-25(x_2-0.7)^2}\right) \tag{2.4.9}$$

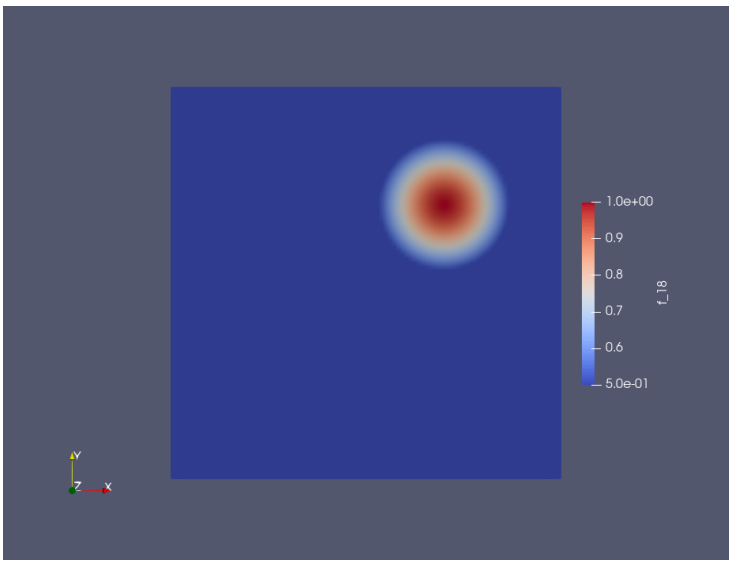


Figure 2.8: Truncated Gaussian blob forcing term

The velocity field $\mathbf{v}$ used for each simulation is the solution of a steady-state Navier-Stokes equation with side walls driving the flow:

$$-\frac{1}{Re}\Delta\mathbf{v} + \nabla q + \mathbf{v}\cdot\nabla\mathbf{v} = 0 \text{ in } \Omega \tag{2.4.10a}$$

$$\nabla\cdot\mathbf{v} = 0 \text{ in } \Omega \tag{2.4.10b}$$

$$\mathbf{v} = \mathbf{g} \text{ on } \partial\Omega. \tag{2.4.10c}$$

The coefficient $Re = 100$ is the Reynolds number, and the Dirichlet term $\mathbf{g}$ is given by $\mathbf{g} = e_2$ on the left wall, $\mathbf{g} = -e_2$ on the right wall and zero everywhere else (see the Advection-Diffusion Bayesian Tutorial in hippylib for more information [131]). The velocity field is shown below.

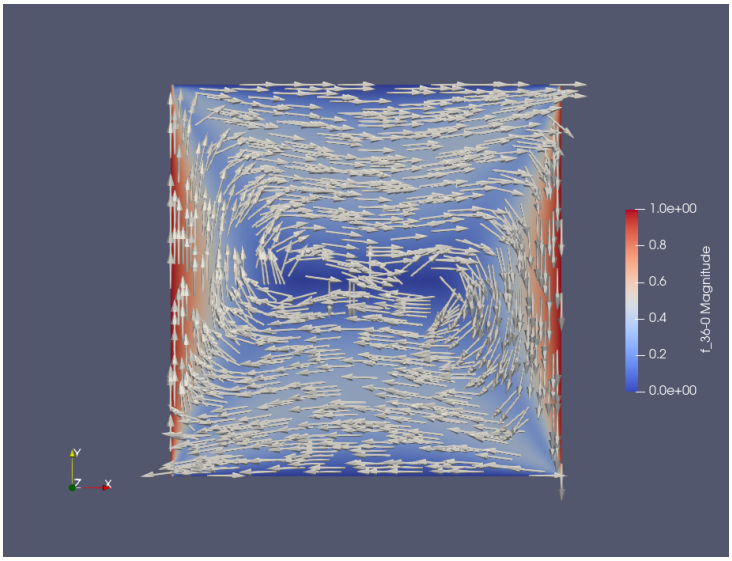


Figure 2.9: Velocity field for convection diffusion PDE

To demonstrate the efficacy of the pLLRN surrogate in a wide variety of contexts, I again consider 16 different parametrizations of the Matern covariance prior for $\gamma, \delta \in \{0.5, 1.0, 2.0, 4.0\}^2$. I consider a sequence of unit square meshes with grid resolution given by $n_x = n_y = 32, 64, 96, 128$, and use piecewise linear finite elements in the solution of the problem. I employ a streamline-upwind stabilization term in the solution of the PDE, which keeps the cell Peclet number bounded by 1. Below I plot samples of the parameter

as well the resulting concentration field u for different parametrizations of the prior.

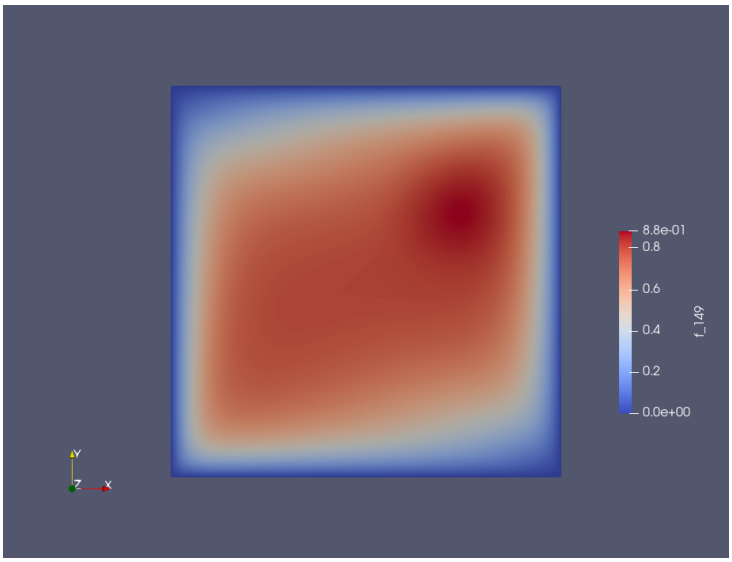


Figure 2.10: The concentration field u for the mean of the uncertain parameter $m = 0$, on $n_x = n_y = 96$

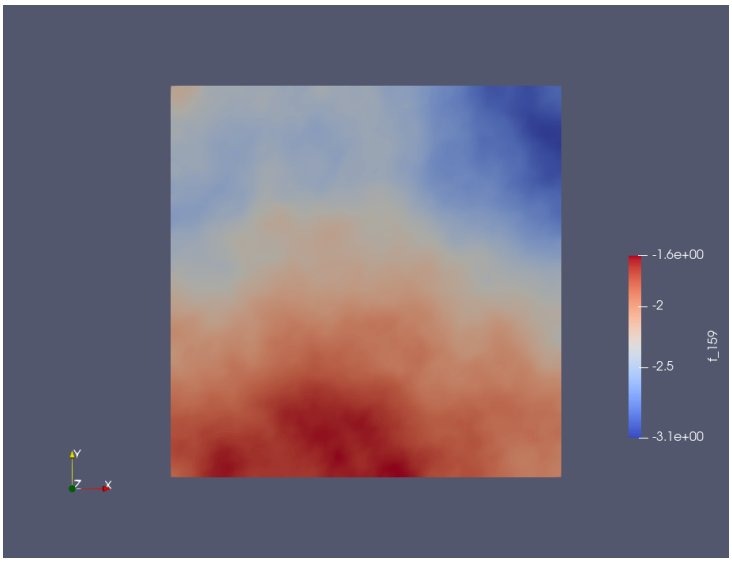


Figure 2.11: A single draw of uncertain parameter m for $\gamma = 0.5$, $\delta = 0.5$

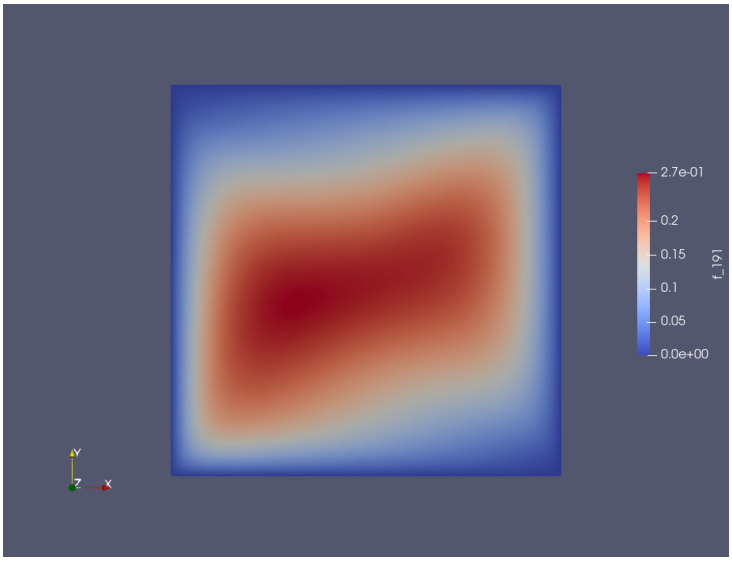


Figure 2.12: The concentration field u evaluated at the sample given sample for $\gamma = 0.5$, $\delta = 0.5$ on $n_x = n_y = 96$

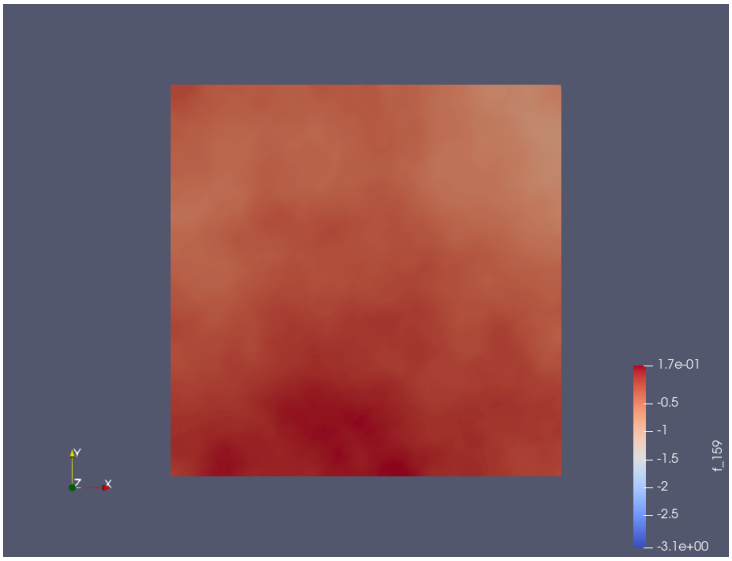


Figure 2.13: A single draw of uncertain parameter m for $\gamma = 0.5$, $\delta = 4.0$

	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 2.0$	$\gamma = 4.0$
$\delta = 0.5$	-14.1	-14.5	-14.7	-14.8
$\delta = 1.0$	-12.8	-12.6	-12.2	-12.0
$\delta = 2.0$	23.8	25.3	26.1	26.5
$\delta = 4.0$	63.8	66.6	68.1	68.7

Table 2.6

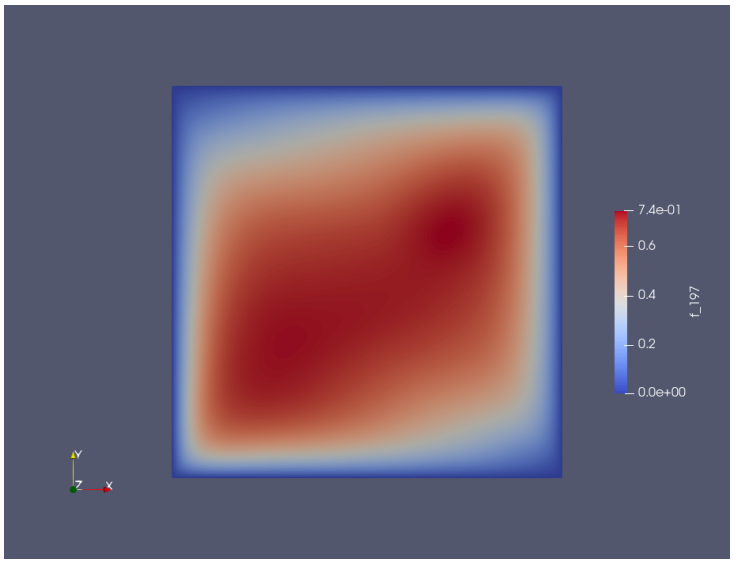


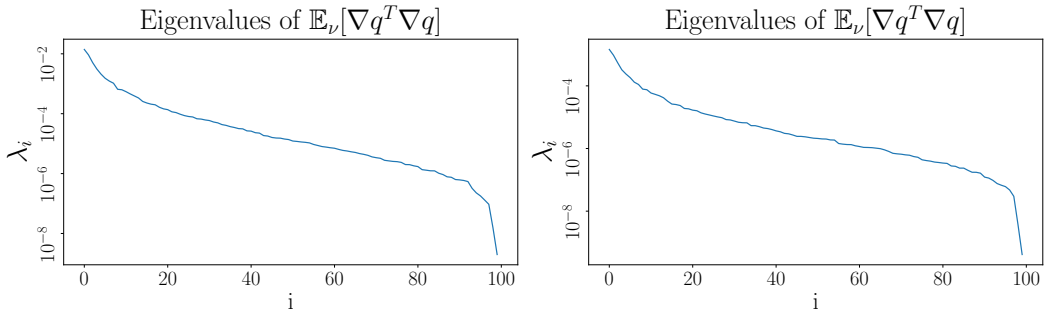
Figure 2.14: The concentration field u evaluated at the sample given sample for $\gamma = 0.5$, $\delta = 4.0$ on $n_x = n_y = 96$

To indicate the inherent difficulty / degree of nonlinearity in the problem below I summarize how a Taylor linear approximation of the map $m \mapsto q$ performed for the different parametrizations of the prior, on the coarsest mesh:

The general trend is that the as γ and δ get larger the problem gets easier. This agrees with intuition about the Matern covariance prior where for a fixed correlation length (controlled by the ratio γ/δ) larger δ decreases the marginal variance. So for a fixed correlation length, larger δ concentrates more of the mass of the prior ν around the mean. For this problem, the Taylor approximation did slightly worse when on the finer meshes so I include the coarsest mesh Taylor approximation, since it could be prolongated onto the

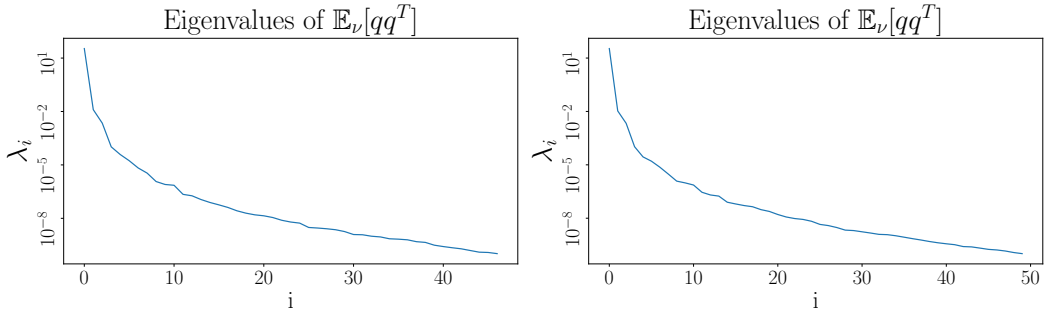
finer meshes.

As with the last example, I study three different projection ranks $r = 20, 40, 80$ for the pLRRNs. The spectral decay of the POD eigenvalues was pretty fast and typically suggested that the rank of the map was near to $r = 40$, although marginally higher for harder problems. The choice of $r = 20$ was chosen because the POD spectrum decayed five to six orders of magnitude in the first 20 eigenvalues across the 16 different problems. The active subspace spectral decay was slower, and suggested the inherent dimensionality of input space was closer to 80. See below for plots of spectral decay and the mesh independence of the spectral properties.



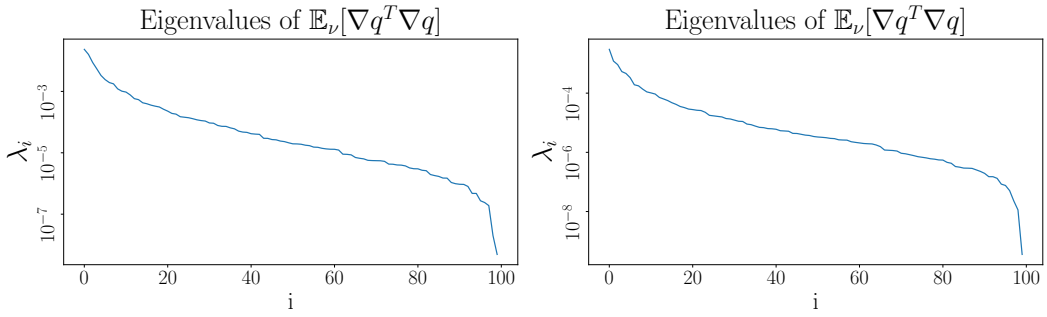
(a) Active subspace input eigenvalue decay for $\gamma = 4.0, \delta = 4.0$ for $n_x = n_y = 64$

(b) Active subspace input eigenvalue decay for $\gamma = 4.0, \delta = 4.0$ for $n_x = n_y = 96$



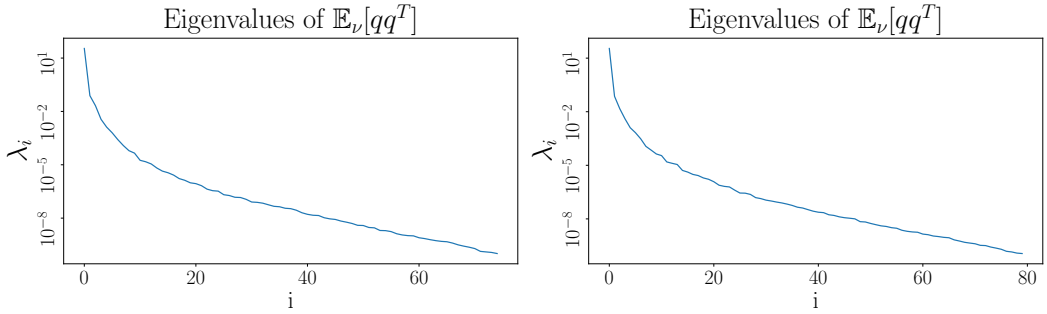
(a) POD eigenvalue decay for $\gamma = 4.0, \delta = 4.0$, for $n_x = n_y = 64$

(b) POD eigenvalue decay for $\gamma = 4.0, \delta = 4.0$, for $n_x = n_y = 128$



(a) Active subspace input eigenvalue decay for $\gamma = 0.5, \delta = 4.0$ for $n_x = n_y = 64$

(b) Active subspace input eigenvalue decay for $\gamma = 0.5, \delta = 4.0$ for $n_x = n_y = 96$



(a) POD eigenvalue decay for $\gamma = 0.5, \delta = 4.0$, for $n_x = n_y = 64$

(b) POD eigenvalue decay for $\gamma = 0.5, \delta = 4.0$, for $n_x = n_y = 128$

For this example the pLRRNs have four nonlinear layers, each with inner layer rank 12. The activation functions used were the same as in the last example, and experimentally performed better than other choices of activation functions that I experimented with. The dense network also has four layers and uses the same activation function structure. Initial guesses for training are sampled from Gaussian white noise : $\mathcal{N}(0, I_{d_W})$, where d_W is the dimension of the neural network weight space.

For the first test I used 800 training data and 200 testing data. At each iteration 200 data are used for the gradient formation and for increased computational economy only 50 data are used for each Hessian vector product. I chose small training sets since this is typical of the target setting where training data construction is by far the dominant cost. Also choosing less training data made it possible to study a wider variety of test problems.

A summary of the dimensions for the different problems is below in the following table:

$n_x = n_y =$	32	64	96	128
d_M	$1,089$	$4,225$	$9,409$	$16,641$
pLRRN $r = 20$	$2,620$	$2,620$	$2,620$	$2,620$
pLRRN $r = 40$	$5,140$	$5,140$	$5,140$	$5,140$
pLRRN $r = 80$	$10,180$	$10,180$	$10,180$	$10,180$
Dense 100	$169,600$	$483,200$	$1,001,600$	$1,724,800$

Table 2.7: Size of configuation spaces and parameter dimension for different meshes.

I summarize the results for all 64 test problems in the following tables. I begin with the problems on the coarsest mesh. The results shown below are for a single optimization run starting from a random initial guess for the neural network weights. The neural network training problem for this PDE example was typically much harder than in the ice sheet problem. In the last case, first order and second order optimizers performed about as well as one another. In this case, first order optimizers did not perform well, and only second order methods inexact Newton CG and low rank Saddle Free Newton performed well on this problem. As in the last example all results reported here are for training results using inexact Newton CG. Since this neural network training problem was very hard, sometimes inexact Newton CG got stuck in parameter space and the resulting testing error is quite bad. This is an artifact of a different problem than what is being studied explicitly in this chapter; that problem is addressed in Chapter 3 of this work. The first numerical results in this section are for one initial guess, and are reported as is. In later numerical tests I report sample statistics for many different initial guesses. The best run for each problem is emboldened.

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.5$	(26.2, 25.7, **27.4**, 18.5)	(27.0, **27.8**, 27.0, 19.7)
$\delta = 1.0$	(**48.5**, 48.1, 48.3, 39.1)	(48.9, 21.8, **50.6**, 45.3)
$\delta = 2.0$	(65.8, 66.4, **67.2**, 56.1)	(67.4, 67.8, **68.4**, 66.7)
$\delta = 4.0$	(74.8, -101, **75.6**, 75.3)	(74.9, 75.2, 76.6, **76.9**)

Table 2.8: Review of neural network testing accuracy results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 32$

	$\gamma = 2.0$	$\gamma = 4.0$
$\delta = 0.5$	(26.6, -10.0, **27.6**, 23.3)	(15.2, 26.9, 27.9, **29.1**)
$\delta = 1.0$	(48.7, 41.5, 49.1, **49.2**)	(**50.0**, 49.4, 46.9, 49.3)
$\delta = 2.0$	(-90., 66.7, 66.5, **68.1**)	(68.0, 66.9, 68.6, **69.0**)
$\delta = 4.0$	(76.4, 77.3, **77.6**, 77.1)	(77.2., 77.4, 77.8, **78.4**)

Table 2.9: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 32$

For the coarsest mesh there is not a clear winner; the pLRRN $r = 80$ performs the best for seven of the problems, the dense 100 neural network performs the best for six of the problems, followed by the PLRRN $r = 20$ performing the best on two of the problems and the pLRRN $r = 40$ performing the best on only one of the problems. Clearly some of the optimization runs got stuck at sub-optimal regions of parameter space, but for the most part the different neural networks tended to perform roughly the same, and all outperformed the Taylor approximations. In this set of examples the most parsimonious network, the pLLRN $r = 20$ performed comparably to the best network in every case except for $\gamma = \delta = 2.0$, which is likely an artifact of the neural network training problem. This indicates that the majority of the information that the neural networks learned during training is in the dominant 20 dimensional subspaces of the input and output as represented by the active subspace and POD reduced bases.

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.5$	(**31.8**, 27.2, 17.3, 20.8)	(**31.1**, 28.1, 26.8, 6.34)
$\delta = 1.0$	(**50.6**, 49.5, 48.5, 43.5)	(**51.1**, 45.9, 46.2, 48.4)
$\delta = 2.0$	(**65.1**, 51.4, 64.4, 61.8)	(66.0, 61.9, 66.3, **66.5**)
$\delta = 4.0$	(74.3, 74.0, 74.6, **76.2**)	(75.2, 75.5, 75.8, **76.5**)

Table 2.10: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 64$

	$\gamma = 2.0$	$\gamma = 4.0$
$\delta = 0.5$	(31.6, **32.1**, 27.1, 21.5)	(**31.9**, 27.2, 23.4, 24.8)
$\delta = 1.0$	(**51.1**, 49.0, 45.4, 45.8)	(**51.6**, 45.8, 41.1, 42.4)
$\delta = 2.0$	(66.0, 66.6, **66.7**, **66.7**)	(**67.2**, 66.6, 66.3, 66.4)
$\delta = 4.0$	(76.0, 76.1, **76.3**, 72.0)	(75.8, 76.3, **76.6**, 76.3)

Table 2.11: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 64$

For the first intermediate mesh ($n_x = n_y = 64$) the neural networks tended to perform better overall, and didn't get stuck at as many poor generalization regions of parameter space. In this set of problems the most parsimonious neural network, the PLRRN $r = 20$ performed the best on nine of the runs. The dense 100 neural network performed the best on four of the problems, the pLRRN $r = 80$ performed the best on three of the problems (tying the dense 100 on one of them), and again the pLRRN $r = 40$ performed the best on only one of the problems. Again the low dimensional configuration spaces that are restricted to the dominant subspaces of the input and output were able to represent the key information of the map as well as, and often better than the discretization dimension dependent network which had dimension $483,200$ in this case.

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.5$	(**38.2**, 37.1, 33.8, 30.5)	(**38.1**, 34.8, 34.3, 18.1)
$\delta = 1.0$	(57.4, 56.5, **55.7**, -43.0)	(**57.8**, 56.8, 56.4, -37.4)
$\delta = 2.0$	(70.2, 68.4, 68.9, **70.3**)	(70.3, 69.0, **71.0**, -28.4)
$\delta = 4.0$	(73.0, 73.3, **74.3**, -34.0)	(75.3, 74.0, **76.6**, -39.3)

Table 2.12: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 96$

	$\gamma = 2.0$	$\gamma = 4.0$
$\delta = 0.5$	(**37.2**, 11.0, 32.2, 22.5)	(**38.8**, 33.4, 18.1, 21.1)
$\delta = 1.0$	(**57.9**, 56.4, 57.5, 49.0)	(**59.0**, 56.5, 58.4, -30.0)
$\delta = 2.0$	(**70.8**, 65.1, 69.5, -51.1)	(69.4, **70.3**, 70.1, -34.6)
$\delta = 4.0$	(**76.6**, 73.5, 74.5, -29.5)	(**76.9**, 76.7, 75.7, 76.0)

Table 2.13: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 96$

As the discretization dimension for the problem increases a trend starts to emerge, showing that the pLRRN networks start to really outperform the neural network that is parametrized by the discretization dimension. In this case the pLRRN $r = 20$ performed the best on ten of the problems, followed by the pLRRN $r = 80$ performing best on four of the problems. The pLRRN $r = 40$ and the dense 100 each performed the best on only one problem, but the pLRRN $r = 40$ was competitive on almost all of the problems, whereas the dense 100 typically had very poor generalization error. In this case the discretization dependent neural network dimension was $d_W = 1,1001,600$. It is clear that the neural network training problem starts to become much more difficult, as there are many more modes that need to be inferred than the data can infer for this overparametrized network. For fixed access to training data the parsimonious projected neural networks performed much better on a per iteration count. This does not take into account also that each training iteration for the overparametrized network becomes much more

expensive. The projected neural networks achieve much better performance for fixed access to data, and also train orders of magnitude faster due to the small configuration dimension.

	$\gamma = 0.5$	$\gamma = 1.0$
$\delta = 0.5$	(**24.1**, 21.1, 22.0, 19.8)	(18.8, 19.1, **19.5**, -45.4)
$\delta = 1.0$	(43.7, 41.4, **44.5**, -94.6)	(**44.8**, 41.8, 43.0, -91.7)
$\delta = 2.0$	(**61.9**, 61.7, 62.1, -101.)	(**63.7**, 60.0, 62.7, -170.)
$\delta = 4.0$	(73.2, 73.2, 73.9, **74.7**)	(72.3, 71.1, **72.8**, -107.)

Table 2.14: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 128$

	$\gamma = 2.0$	$\gamma = 4.0$
$\delta = 0.5$	(**23.4**, 14.8, 20.1, 19.6)	(17.8, 15.3, **19.6**, -50.0)
$\delta = 1.0$	(**44.8**, 43.0, 43.6, -105.)	(**46.9**, 45.5, 44.6, -8.13)
$\delta = 2.0$	(64.4, 60.9, **65.0**, -67.0)	(64.0, 59.2, **65.2**, -60.7)
$\delta = 4.0$	(**72.4**, 71.8, 72.3, -98.8)	(**71.9**, 67.0, 71.7, -162.)

Table 2.15: Review of neural network training results for prior parameter sweep (pLRRN $r = 20$, pLRRN $r = 40$, pLRRN $r = 80$, Dense 100) for mesh $n_x = n_y = 128$

The trend continues for the finest mesh ($n_x = n_y = 128$), where the configuration space dimension grows to $d_W = 1,724,800$. The most compact neural network, the pLRRN $r = 20$ is again the winner as it performs the best on nine of the problems. The pLRRN $r = 80$ performs the best on six of the problems, and the dense 100 network performs the best on one of the problems. The pLRRN $r = 40$ network was again competitive with the other networks in most of the problems, but never the best. The dense 100 network was largely uncompetitive, and typically got stuck in regions of configuration space with very poor generalization error, demonstrating how the neural network training problem gets harder as the intrinsic dimension becomes very small relative to the discretization dimension.

This sequence of results demonstrates the need for neural network architectures that are discretization dimension independent. As the discretization dimension grows the conventional strategy of training a neural network based on the input and output dimension performs worse. The use of dominant subspace information led to neural networks that performed about the same on the four different problems. For each different resolution of the problem, these projected neural networks are only attempting to resolve the mapping $m \mapsto q$ in informed subspaces of the parameter that are the same dimension for each mesh used. When the discretization dimension dependent strategy performed well, it did not drastically outperform the projected strategy. This suggests that when the discretization dependent networks perform well they are learning the representation of the mapping in the dominant data informed subspaces of the input and output.

In what follows I focus on a two example problems from the second to largest mesh. In these two problems I give each neural network ten different initial guesses and report sample statistics for the runs. I allow the projectors used in the pLRRNs to be trained, and study the effect of using the active subspace and POD basis vectors as initial guesses vs random weights. The results are summarized in the tables below. The best generalization accuracies are emboldened.

	Dimension	Max accuracy	Mean accuracy
pLRRN $r = 20$	2,620	71.58	70.33
pLRRN $r = 40$	5,140	73.56	68.85
pLRRN $r = 80$	10,180	70.13	68.86
pLRRN Train $r = 20$	192,800	**74.19**	70.30
pLRRN Train Random $r = 20$	192,800	68.54	26.39
pLRRN Train $r = 40$	385,500	72.50	**70.31**
pLRRN Train Random $r = 40$	385,500	74.07	-155.63
pLRRN Train $r = 80$	770,900	73.21	69.12
pLRRN Train Random $r = 80$	770,900	72.25	-275.59
Dense 100	1,001,600	16.84	-40.57

Table 2.16: Comparison of different training strategies for $\gamma = 2.0$, $\delta = 2.0$, $n_x = n_y = 96$ over 10 different initial guesses

	Dimension	Max accuracy	Mean accuracy
pLRRN $r = 20$	2,620	76.08	72.97
pLRRN $r = 40$	5,140	75.68	74.79
pLRRN $r = 80$	10,180	76.63	74.85
pLRRN Train $r = 20$	192,800	79.61	**78.73**
pLRRN Train Random $r = 20$	192,800	79.52	58.96
pLRRN Train $r = 40$	385,500	79.20	76.81
pLRRN Train Random $r = 40$	385,500	79.23	-104.93
pLRRN Train $r = 80$	770,900	78.30	75.20
pLRRN Train Random $r = 80$	770,900	**79.76**	-372.59
Dense 100	1,001,600	78.20	-3.90

Table 2.17: Comparison of different training strategies for $\gamma = 1.0$, $\delta = 4.0$, $n_x = n_y = 96$ over 10 different initial guesses

In both cases the dimension independent neural networks perform consistently well. Their maximum performance over the ten runs is close to their average performance. Allowing the projectors to be trained does increase the accuracy, but it comes at a cost of increasing the configuration space by several orders of magnitude.

Interestingly the neural networks with random projectors used as initial guesses in their best runs can perform comparable to the ones using the active subspace and POD projectors as initial guesses. A random initial guess for the projector embeds the input and output data in rank $r = 20, 40, 80$ subspace that is likely to capture dominant information of the data with high probability / in expected value; this is the so called Johnson-Lindenstrass transform [2, 27]. Since the active subspace and POD projectors are computed via randomized sampling of the matrices, the approximation of the dominant subspace must lie in the span of the Gaussian matrices used in the randomized computation of the bases. For this reason it makes sense that random initial guesses for the projectors can find configurations that are comparable to the "optimal" initial guesses given by active subspace and POD[1]. What is clear however is that these random initial guesses do not do a

[1]However randomized embedding of the input data is related to the Karhunen-Loève expansion, not the active subspace, which has a tighter error bound.

good job of resolving the dominant modes of the map *on average*. Similarly the dense 100 network is able to find on good configuration in ten runs for the latter problem $\gamma = 1.0$, $\delta = 4.0$ but has poor performance on average. For the former problem, $\gamma = 2.0$, $\delta = 2.0$ the dense 100 network was able to find no configurations that performed better than 16.87% accuracy on testing data.

These tables show that the pLRRN architecture is effective in resolving the mapping $m \mapsto q$, and having optimal initial guesses for the projector based on active subspace and POD leads to much better approximation on average over initial guesses. Since the computation of these dominant subspaces can be efficiently computed via adjoint methods (linearizations of the map, no nonlinear solvers) and randomized methods these optimal initial guesses should be used if possible.

Next I study the effect of data training population and depth on the approximation capabilities for the projected low rank residual network. First for the $\gamma = 2.0, \delta = 2.0$ problem on the $n_x = n_y = 96$ mesh, I study the effect of training data size. The goal of neural network training, *generalization*, has to do with how well our statistical model of the problem (via Monte Carlo subsampling) approximates the true problem. Classically this Monte Carlo error is bounds the approximation error in expectation with respect to partitions of the data by an upper bound for the elementwise standard deviation of the data divided by the square root of the number of samples used. This means that one can hope to incur an approximation error of the true problem of order $O(N_{\text{data}}^{-\frac{1}{2}})$ in expectation with respect to data sampling. This is among other approximation errors, which makes it difficult to draw direct experimental conclusions about approximation error incurred in subsampling, but trends can be observed.

For this problem I instantiate 50 different partitions of the whole training population $X_k \subset X$, and run training on problems with $|X_k| = 200, 1000, 1800$ and report sample statistics for generalization error below in the following

table.

	Max accuracy	Mean accuracy	Median accuracy
pLRRN $\|X_k\| = 1800$	72.23	69.33	69.84
pLRRN $\|X_k\| = 1000$	70.15	66.67	67.67
pLRRN$\|X_k\| = 200$	68.15	53.45	49.92

Table 2.18: Data partitioning approximation error taken over 50 data partitions for $\gamma = 2.0$, $\delta = 2.0$, $n_x = n_y = 96$, pLRRN $r = 40$

These results agree with the classical result that in expected value the approximation should improve as the training data population increases. The maximum generalization error and median also obey this trend: as more data is used, the approximation improves for these statistics as well.

For the last set of numerical results I investigate the effect of neural network depth on the approximation error for the problem $\gamma = 2.0$, $\delta = 2.0$ on the coarsest mesh $n_x = n_y = 32$. Again I report sample statistics taken over ten different initial guesses for the neural network weights.

	Max accuracy	Mean accuracy	Dimension
pLRRN $r = 40$, depth $= 1$	66.88	66.88	2,140
pLRRN $r = 40$, depth $= 2$	67.15	67.15	3,140
pLRRN $r = 40$, depth $= 3$	67.83	67.36	4,140
pLRRN $r = 40$, depth $= 4$	67.46	66.91	5,140
pLRRN $r = 40$, depth $= 5$	67.59	67.00	6,140
pLRRN $r = 40$, depth $= 6$	67.76	67.33	7,140
pLRRN $r = 40$, depth $= 7$	67.77	67.31	8,140
pLRRN $r = 40$, depth $= 8$	67.83	66.89	9,140
pLRRN $r = 40$, depth $= 9$	67.87	66.81	10,140
pLRRN $r = 40$, depth $= 10$	68.21	67.07	11,140

Table 2.19: Depth approximation error for projected low rank resnet architecture with $r = 40$ taken over 10 initial guesses for $\gamma = 2.0\delta = 2.0, n_x = n_y = 32$

The general trend is that the maximum accuracy tends to increase as the depth of the neural network increases. However, the average performance does suffer for the deeper networks. This agrees with the experimental observation that neural networks become harder to train as the depth increases.

As the neural network becomes deeper, and the configuration space becomes larger, it may have more approximation power, but finding generalizable local minima may also become harder. This is a difficult trade-off in neural network architecture construction.

2.5 Conclusion

Many-query / outer loop applications involving parametric maps require surrogates to be tractable at large scales. In such settings a mapping from a model parameter m to an output quantity of interest q must be evaluated many times. Each evaluation requires the evaluation of an expensive high dimensional model. For many parametric mappings of this sort, the quantity of interest are informed in only few modes of the input and output subspaces. The dimension of this dominant mode representation of the mapping is referred to as the intrinsic dimensionality of the map.

Data-driven approaches that have become very popular in recent years will suffer as the discretization dimension grows for the underlying model used in the parametric map evaluation. As the discretization dimension grows, the information content for the map does not necessarily grow, and the large dimensional models become harder and harder to train.

In this work I analyze and propose strategies for neural network construction that use model information to restrict the neural network architecture to the dominant subspaces of the input and output spaces as exposed by active subspace and POD. In settings where the spectra for the active subspace and POD operators decay quickly, the map can be well approximated in dominant subspaces of the input and output spaces.

I propose neural network parametrizations between these dominant subspaces as a strategy for creating discretization dimension independent surrogate models. When parametric maps involve expensive high dimensional models, the generation of training data is the main bottleneck. Having a parsimoniously parametrized model that has good approximation capability

is important in settings where large training data sets are infeasible to generate, since one typically wants commensurate training data cardinality to number of parameters to be inferred. The typical data-driven approach has a configuration space that scales with the discretization dimension (which is typically several orders of magnitude larger than the intrinsic dimension), but for expensive models often it is only possible to generate training data sets that are on the order of hundreds or thousands of pair. Identifying discretization dimension independent low dimensional structure is critical for parametrizing models in this context. This is because one can only hope to infer a number of parameters for a surrogate that is commensurate to the number of training data available. For applications of interest, only small training data sets are feasible to generate at large scales.

In order to make the problems discretization dimension independent, dominant subspaces of the input and output can be computed offline using efficient adjoint methods, randomized linear algebra and Monte Carlo approximations. The individual operations used in constructing these subspaces are in principle much cheaper than the evaluation of the nonlinear map itself, since they involve linearizations. If the rank of the active subspace operator decays fast, the marginal cost of the subspace approximation can be made cheaper than the nonlinear evaluation of the map itself.

The important information content for the parametric map lie in these dominant subspaces. Restricting models to only resolving these modes serves as a form of regularization. Since the orthogonal complements to these subspaces do not inform the input-output map as much, learning these modes is not as important. Restricting to these dominant subspaces can help avoid the surrogate model learning noise in the input-output map, which can happen when the surrogate is overparametrized.

I propose a sparsified low rank residual neural network structure as an added means of making the model parsimonious. the configuration space dimension for this model scales linearly with the intrinsic dimension for the

parametric map. The prefactors in the scaling constant involve the depth of the neural network and the rank used for the low rank residual network. I term this neural network strategy projected low rank residual neural networks.

Numerical experiments with two PDE models suggest that models parametrized parsimoniously by the intrinsic dimension of the problem perform as well as data-driven approaches do in the cases where the discretization dimension is not too large. The parsimonious models in this cost often perform marginally better and for orders of magnitude smaller configuration spaces.

As the discretization dimension grows, the data-driven approaches perform worse and worse. The configuration space dimension for the dimension independent neural networks stay the same, the approximation improves in some cases. These numerical results demonstrate the need for discretization dimension independent approaches for creating parametric map surrogates in high dimensions. The performance of the projected neural network surrogates over a range of problems demonstrates how model based information can be useful to help construct useful and scalable neural network surrogates in parametric map surrogate construction.

This work establishes that representing the parametric mapping in dominant modes of the subspace is a principled scalable dimension independent approach for parametric mappings involving PDEs. As I continue this work, I will revise the parametrization of the nonlinear surrogate. The low rank residual network structure seemed successful in the two application problems that I explored. Much can be done to improve the parametrization by exploring different neural network representations as well as initial guesses for the weights, which optimization is highly sensitive to. Other forms of dimension reduction can also be investigated, for example when discretized input data has spatial structure, convolution operations can be useful for compressing information. Methods for finding convolutional kernels in a way that is mesh independent and based on key model structure could be of great use in this

setting.

The approach outlined in this chapter can be easily extended to efficient methods for also training derivative structure. This idea is explored briefly in Appendix 2, along with an idea for a dominant derivative subspace approximation. In future work I hope to apply these ideas to much larger dimensional problems, since the approach could be especially useful in very high dimensional settings where many-query / outer-loop applications are prohibitive. I am planning on applying these ideas to large scale models involing ocean science (MITgcm) [62, 49, 51, 133], large scale ice sheet codes [65, 66], and airfoil design optimization [99].

This work is related to an open question in scientific machine learning: how to use model based information in neural network architecting and training for scientific problems. Various approaches involving making sure the neural network obeys the PDE / conservation laws have been studied [90, 91, 108, 109]. In this work a model constraint is imposed by restricting the neural network to informed subspaces of the input and output. While this strategy doesn't ensure that the governing equations of the model are exactly obeyed, this strategy is a principled way to restrict surrogates to only representing modes of the model that you care about.

Chapter 3

Newton Methods for Stochastic Nonconvex Optimization

I consider the stochastic nonconvex optimization problem

$$\min_{w \in \mathbb{R}^{d_W}} F(w) = \int \ell(f(x,w),y)\, d\nu(x,y), \tag{3.0.1}$$

where ℓ is a smooth (loss) function that measures the error between an approximant $f(x,w)$, and a true mapping $y(x)$[1]. The weight, $w \in \mathbb{R}^{d_W}$ is the vector of optimization variables, the *data pairs* (x,y) are distributed with joint probability distribution $\nu(x,y)$, and $F : \mathbb{R}^{d_W} \to \mathbb{R}$ is referred to as the *expected risk*. This problem arises in machine learning, where the goal is to reconstruct a mapping $x \mapsto y$ with an approximant f, i.e. a deep neural network or other model parametrized by w. See for example [57]. In practice, complete information about ν is not available. Rather, one has access to samples $x_i, y_i \sim \nu$, which leads to the Monte Carlo approximation of (3.0.1) as

$$\min_{w \in \mathbb{R}^{d}} F_X(w) = \frac{1}{N_X} \sum_{i=1}^{N_X} F_i(w), \tag{3.0.2}$$

where $F_i(w) = \ell(f(x_i,w),y_i)$, and $X = \{(x_i,y_i)|x,y \sim \nu\}_{i=1}^{N_X}$. The function $F_X : \mathbb{R}^{d_W} \to \mathbb{R}$ is referred to as the *empirical risk*. Optimization problem (3.0.2) is typically solved via an iteration of the form

$$w_{k+1} = w_k + \alpha_k p_k, \tag{3.0.3}$$

[1]This chapter contains content from [96] (Thomas O'Leary-Roseberry, Nick Alger, Omar Ghattas, Inexact Newton Methods for Stochastic Nonconvex Optimization with Applications to Neural Network Training. arXiv preprint, arxiv:1905.06738, 2019) and [97] (Thomas O'Leary-Roseberry, Nick Alger, Omar Ghattas, Low Rank Saddle Free Newton: Algorithm and Analysis arXiv preprint, arxiv:2002.02881, 2020)

where p_k is typically a gradient based search direction and α_k is the step length (or learning rate as it is known in machine learning). If $p_k = -\nabla F_X(w_k)$, then the iteration is gradient descent. If $p_k = -\nabla^2 F_X(w)^{-1}\nabla F_X(w)$, then the iteration is Newton's method. For many machine learning representations, the cost of evaluating F_X is $O(d_W N_X)$. Computing the gradient using adjoint methods requires $O(d_W N_X)$ work and $O(d_W)$ storage, while explicitly forming the Hessian matrix requires $O(d_W^2 N_X)$ work and $O(d_W^2)$ storage, factorizing it requires $O(d_W^3)$.

Several features make optimization problem (3.0.2) difficult to solve:

1. Large parameter dimension, d_W
2. Large data dimension, $N_X = |X|$
3. Nonconvexity of F_X
4. Ill-conditioning of F_X
5. Stochasticity in ν

Features 1 and 2 make optimization expensive. To ease the computational burden associated with large data dimension, and motivated by the redundancy in the data for large N_X, one typically uses subsamples of data, X_k from X, at each iteration. At each iteration, a subsample data set $X_k \subset X$ is substituted for X in the empirical risk minimization (Equation 3.0.2). This is known as stochastic approximation (SA); when ones uses all of the data X, this is referred to as sample average approximation (SAA). In order to make matrix-free Newton methods more computationally economical a further degree of subsampling may be used for the Hessian matrix, that is, at an iteration k, $S_k \subset X$ is used to approximate the stochastic Hessian matrix, with $N_{S_k} \leq N_{X_k}$. In what follows X_k refers to the data used in the stochastic gradient computation at iteration k, and S_k refers to the data used in the

Hessian computation. That is

$$\nabla F_{X_k} = \frac{1}{N_{X_k}} \sum_{i=1}^{N_{X_k}} \nabla_w \ell(f(x_i, w), y_i) \tag{3.0.4a}$$

$$\nabla^2 F_{S_k} = \frac{1}{N_{S_k}} \sum_{i=1}^{N_{S_k}} \nabla_w^2 \ell(f(x_i, w), y_i) \tag{3.0.4b}$$

Features 3 and 4 create essential difficulty for the solution of the optimization problem. Nonconvexity makes it computationally intractible (NP-hard) to find global minima [13, 93], and riddles the energy landscape with saddle points where gradient based iterates can get stuck [125]. Ill-conditioning makes typical gradient based methods slow to converge[7, 78, 81, 115].

Feature 5 leads to an issue of generalizability of a candidate solution w^*. Candidate optima w^* obtained from solution of empirical risk minimization problems are judged by how well they perform on unseen testing data. Because of sampling error, a solution to (3.0.2) may be a poor approximation of a solution to (3.0.1). It is desirable to devise optimization schema that do not overfit to the noise of the training data used to train them, if possible.

It is a common belief in the machine learning community that Newton methods are prohibitive due to the computational costs of forming and inverting the Hessian matrix arising in empirical risk minimization. This widely held belief has made first order optimizers the method of choice in many settings. That aside, there has been interest in the application of second order methods to this stochastic nonconvex regime. Recent work involving second order methods in this setting has focused on two major themes:

1. How stochastic second order information can be used to improve convergence over first order methods.

2. How information about indefiniteness can be used to facilitate evasion of saddle points and indefinite regions of parameter space.

In the first theme, approximations of Hessian based curvatures can be obtained efficiently using methods based on the Gauss-Newton Hessian to

efficiently approximate the Newton solve [26, 83, 86, 84]. Recent work advocates for the use of the Gauss-Newton Hessian in the stochastic setting [53].

Methods that use the full stochastic Hessian (which has indefiniteness) have been shown to converge rapidly when the Hessian is subsampled more than the gradient, because the variance of the Hessian is typically smaller than that of the gradient [45, 111, 112, 134]. Along with this push, Roosta et al. explore stochastic inexact Newton methods, and derive probabilistic bounds for spectral convergence of the subsampled Hessian to the true Hessian [111, 112, 135, 136]. In addition, they experimentally demonstrate that while Gauss-Newton methods may help with the conditioning of the optimization problem, they are prone to getting stuck at saddle points.

Bollapragada et al. analyze inexact Newton conjugate gradient (CG) methods in the semi-stochastic setting (where the Hessian is subsampled, but not the gradient) for convex problems [15]. In these methods, CG iterations for solving the Newton system are terminated when a coarse solver tolerance is met, thereby reducing the complexity to $O(kdN_X)$ operations, where k is the number of CG iterations, which depends on the clustering of the eigenvalues of the Hessian.

For the second theme, the focus is on understanding how optimizers perform in the vicinity of saddle points. Nonconvex energy landscapes typically contain many strict saddle points (stationary points with at least one direction of negative curvature), as well as local minima that may not be global minima. How best to deal with saddle points is an open area of research. Some work has been done to classify when nonconvex problems are tractable [125].

In deep learning, it is conjectured that local minima are almost as good as global minima [36]. However, finding local minima is still difficult because the energy landscape is riddled with saddle points [41]. Saddle points typically correspond to suboptimal solutions in nonconvex optimization problems such

as matrix factorization and phase retrieval [67, 126].

Significant work has been dedicated to understanding how first order methods perform in the vicinity of strict saddle points [54, 68, 75, 76]. Saddle points slow the local convergence of first order methods (methods in which p_k is constructed using only gradient information). These methods typically escape saddle points asymptotically. Reddi et al. argue for using first order methods while the gradient is large and switching to second order methods when near stationary points [110]. Jin et al. argue for adding noise uniformly sampled from a ball with radius large enough to dominate saddle regions, where optimizers get stuck [68].

Newton's method without modification converges locally to strict saddle points, since gradient components initially oriented away from the saddle are reoriented towards the saddle point due to the associated negative eigenvalue of the Hessian. In numerical optimization, modified Newton methods that enforce positive definiteness of the Hessian (for example by maintaining positive eigenvalues in a spectral decomposition) are employed to ensure descent for nonconvex problems [56, 95]. One approach that facilitates fast escape from saddle points involves replacing the Hessian with the absolute value of the Hessian, $|H|$ [56]. Specifically, let the spectral decomposition of the Hessian be given as follows:

$$H = U\Lambda U^T = \sum_{i=1}^{d} \lambda_i u_i u_i^T, \tag{3.0.5}$$

where the eigenvalues λ_i are sorted such that $|\lambda_i| \geq |\lambda_j|$ for all $i > j$, and $u_i \in \mathbb{R}^d$ are the associated eigenvectors. The absolute value of the Hessian thus is

$$|H| = \sum_{i=1}^{d} |\lambda_i| u_i u_i^T.$$

Then $|H|$ is used in place of H within the Newton system. In the vicinity of a saddle point, the gradient components in negative definite direction point away from the saddle point. When the normal Hessian inverse is applied to

the gradient, these components are flipped toward the saddle point. When the inverse of $|H|$ is applied the components remain pointed away from the saddle point, and are optimally rescaled by the local curvature in that direction. The indefiniteness of the Hessian matrix must come from the non Gauss-Newton portion (the Hessian minus the Gauss-Newton Hessian), so methods based on the Gauss-Newton Hessian will not be able to facilitate fast escape in this way.

In machine learning, the use of the spectral decomposition of the Hessian with absolute values of eigenvalues was introduced by Dauphin et al. under the name of saddle free Newton (SFN) [41]. This method uses a Krylov procedure (Lanczos) to form an approximation of $|H|$. Computing the Lanczos approximation of order r does not require forming or factorizing the Hessian matrix; instead only the application of the Hessian to r vectors is required. Paternain et al. prove that a variant of the SFN algorithm converges to ϵ-local minima (where the gradient norm is less than ϵ) with probability $1 - p$ in $O(\log(1/p)) + O(\log(1/\epsilon))$ iterations [103].

In this work I investigate the use of matrix-free inexact Newton methods in the solution of stochastic nonconvex optimization problem.

I start by investigating stochasticity in the spectral structure of stochastic indefinite operators, such as Hessians in stochastic nonconvex optimization problems. A statistical model for a stochastic indefinite operator shows that variance in certain modes of stochastic operators can become unbounded near inflection points where the Hessian transitions from being local indefinite to locally semi-positive definite in these directions. This analysis suggests that there is a link between indefiniteness and high-variance eigenvalues of the stochastic Hessian used in empirical risk minimization. High variance (noisy) modes very often are not informative of the statistical relationship that is being learned, and are due to stochastic variations specific to training data. This finding is observed in numerical examples, where smaller eigenvalues of subsampled Hessians exhibit large variance when the operator is indefinite.

Large eigenvalues do not exhibit as much variance, and are also less likely to be close to an inflection point if the Hessian is not varying too rapidly.

In empirical risk minimization problems, overfitting occurs when information specific to training data instances, that are not representative of the true distribution (i.e. noise) are learned during training. It is desirable avoid incorporating noisy Hessian information during training to guard against overfitting. In this work I attempt to analyze second order methods based on what spectral information they attempt to resolve, and how it could be dominated by noise and lead to poor performance.

I investigate stochastic inexact Newton-Krylov methods and their extension to the nonconvex regime. Newton-Krylov methods are optimal in the sense that they use information about the entire Hessian spectrum to approximate the Newton direction, unfortunately this includes noisy modes of the Hessian. This is a potential drawback for the use of Krylov processes in the stochastic nonconvex setting. While the spectral collapse of the stochastic nonconvex Hessian may be amenable to Krylov procedures, Krylov procedures cannot avoid paying attention to noisy modes, which may incorporate information that leads to overfitting. This can be remedied to some extent via the use of early termination procedures.

This key issue of noise motivates the novel algorithm Low Rank Saddle Free Newton (LRSFN), which employs a low rank approximation of the Hessian that resolves the dominant modes of the Hessian. These modes are likely to be have low variance. Randomized linear algebra can give accurate approximations of low rank eigenvalue decompositions of Hessians for only a few Hessian vector products. Since these modes are low variance they can be approximated for few samples making the method highly economical. Like the original saddle free Newton algorithm (SFN) [41], LRSFN uses the absolute value of the eigenvalues of the Hessian to facilitate fast escape from indefinite regions. Unlike SFN, which uses a Krylov procedure to approximate the Hessian spectrum, LRSFN is designed to only resolve high certainty modes

of the stochastic nonconvex Hessian. This leads to both better generalization and better convergence properties.

Extending the work of [15], I prove local convergence rate bounds for inexact Newton Krylov methods and LRSFN. These bounds help quantify the how local convergence is sensitive to subsampling the gradient and Hessian, approximating the Newton linear solve, and hyperparameters such as regularization parameter and step length α_k. These methods can achieve fast convergence in the vicinity of a local minimum.

I demonstrate the performance of these methods on standard machine learning problems (MNIST classification and CIFAR10 autoencoder training), as well as regression problems arising in parametric map surrogate construction. I compare the methods against standard first order methods Adam, gradient descent (GD), stochastic gradient descent (SGD), and the SFN method [40, 46]. Numerical results show that LRSFN outperforms all of the other methods in terms of generalization error, with INCG using early termination as a close second. I investigate the spectral properties of the stochastic nonconvex Hessian: both the spectral decay as well as the noise in the Hessian. These experimental observations agree with other works that demonstrate the fast decay of the Hessian in certain settings and persistent indefiniteness during large phases of neural network training[3, 55, 117]. They also provide empirical evidence for the link between indefiniteness and noise as predicted by analysis.

I hope that this work can demonstrate the computational economy and good performance that can be obtained by efficiently designed second order methods. The LRSFN algorithm with globalization does not require hyperparameter tuning and on numerical experiments is shown to outperform first order methods that require a great deal of hyperparameter tuning. Moreover the inexact Newton CG algorithm with early termination is also demonstrated to be a useful algorithm, with very good convergence properties, although more prone to overfitting than LRSFN. Properties of the Hessian

spectrum (decay and noise), can be leveraged to create efficient second order methods that can lead to better generalization properties.

3.1 Background

Notation: For matrices A and B, $A \succeq B$ means that $A-B$ is semi positive definite. I focus finite dimensional Hilbert spaces with inner product $x^T y$, and corresponding norm $\|\cdot\| = \|\cdot\|_2$, or the Euclidean ℓ^2 distance on vectors in $\mathbb{R}^{d_W}$[2]. By $\mathbb{E}$ I mean the expectation taken against the measure ν. The operator $\mathbb{E}_k$ stands for the conditional expectation at an iteration k taken over all possible sample batches X_k. In the Euclidean space $\mathbb{R}^{d_W}$ denote by $B_r(w)$ the ball of radius r centered at w.

In solving (3.0.2) one seeks to find a candidate solution w^* that satisfies optimality conditions.

Definition 3.1 (Stationary points)**.** *A point w^* is a first order stationary point if $\|\nabla F(w^*)\| = 0$. A point w^* is an ϵ-first order stationary point if $\|\nabla F(w^*)\| < \epsilon$. A point w^* is a second order stationary point if*

$$\|\nabla F(w^*)\| = 0 \quad \textit{and} \quad 0 \preceq \nabla^2 F(w^*). \tag{3.1.1}$$

A point w^* is an (ϵ_g, ϵ_H)-second order stationary point if

$$\|\nabla F(w^*)\| \leq \epsilon_g \quad \textit{and} \quad -\epsilon_H I \preceq \nabla^2 F(w^*), \tag{3.1.2}$$

for some $\epsilon_g, \epsilon_H > 0$. A point w^* is a stochastic (ϵ_g, ϵ_H)-second order stationary point if

$$\mathbb{E}[\|\nabla F(w^*)\|] \leq \epsilon_g \quad \textit{and} \quad -\epsilon_H I \preceq \mathbb{E}[\nabla^2 F(w^*)]. \tag{3.1.3}$$

I seek to solve the empirical risk minimization (3.0.2) via the gradient based iteration (3.0.3), approximately solving the Newton system

$$\nabla^2 F_{S_k}(w_k) p_k = -\nabla F_{X_k}(w_k). \tag{3.1.4}$$

[2]Here I mean the finite dimensional vector space, not the loss function also denoted ℓ

The subsampled gradient is calculated over data X_k, while for computational economy the subsampled Hessian is calculated over data $S_k \subset X_k$, where $N_{S_k} \ll N_{X_k}$. Due to ill-conditioning and nonconvexity I consider the Tikhonov regularized empirical risk minimization problem

$$\min_w \overline{F}_X = \frac{1}{N_X} \sum_{i=1}^{N_X} F_i(w), + \frac{\gamma}{2} \|w\|^2, \tag{3.1.5}$$

for some $\gamma > 0$ [128]. Ill-posedness of the neural network training problem is investigated in Appendix 1. Much of the following can be extended to other regularizations such as ℓ^1 or cubic regularization, however this is out of the scope of this work.

I now state some assumptions that will be used later in the work. Assumptions A1-A4 are adapted from [15].

A1 (Dominant positive eigenvalues) The function F is twice continuously differentiable and any subsampled Hessian is spectrally bounded from above with constant L. That is, for any integer N_S and set S with $|S| = N_S$, there exists a positive constant $L_{N_S} < L$ such that

$$\nabla^2 F_S(w) \preceq L_{N_S} I. \tag{3.1.6}$$

Moreover the first r eigenvalues of $\nabla^2 F_S(w)$ evaluated along a path of iterates starting at w_0 are positive.

A2 (Bounded variance of sample gradients) There exists a constant v such that

$$tr(\text{Cov}(\nabla F_i(w))) \leq v^2 \quad \forall w \in \mathbb{R}^d \tag{3.1.7}$$

A3 (Lipschitz continuity of Hessian) The Hessian of the objective function F is Lipschitz continuous, i.e., there exists a constant $M > 0$ such that

$$\|\nabla^2 F(w) - \nabla^2 F(z)\| \leq M \|w - z\|^2 \quad \forall w, z \in \mathbb{R}^d \tag{3.1.8}$$

A4 (Bounded variance of Hessian components) There exists σ such that, for all component Hessians,

$$\|\mathbb{E}[(\nabla^2 F_i(w) - \nabla^2 F(w))^2]\| \leq \sigma^2, \quad \forall w \in \mathbb{R}^d \tag{3.1.9}$$

A5 (ϵ_g-first order stationary point). For a given candidate stationary point w^* and gradient batch size N_{X_k}, there exists $\epsilon_g > 0$ such that

$$\mathbb{E}_k[\|\nabla F_{X_k}(w^*)\|] \leq \epsilon_g \tag{3.1.10}$$

3.2 Noise in stochastic indefinite Hessians

Here, variance in the spectra of subsampled Hessians arising in stochastic nonconvex optimization problems is investigated. Analysis focuses on the statistical quantity

$$\text{var}_{S_k}\left(\frac{u_i^T[\nabla^2 F_{S_k}]u_i}{u_i^T u_i}\right), \tag{3.2.1}$$

that is, the subsampling variance of the Rayleigh quotients with respect to each eigenvectors u_i of the true expected risk Hessian $\nabla^2 F$. This statistical quantity represents uncertainty in the subsampled eigenvalue approximation. When the variance is high, subsampled Hessian eigenvalue approximations in these directions are likely to be highly inaccurate, i.e. statistically noisy. This variance is calculated with u_i fixed against against different subsampled Hessian matrices $\nabla^2 F_{S_k}$, with the same cardinality for each subspace S_k. When this quantity is large for a particular direction u_i, it means that the direction is dominated by noise. If search directions are heavily influenced by directions dominated by noise, this can lead to poor generalization and overfitting. If one can avoid incorporating noisy information into an optimization scheme, one can intuitively observe better generalization properties.

In this section a Gaussian statistical model is used to investigate Rayleigh quotient variance. This model is consistent with the true expected risk Hessian asymptotically (in expectation). For this model, Rayleigh-quotients can exhibit high variance in regions of indefiniteness, and when eigenvalues of

the Hessian change sign between different batches. This agrees with numerical results observed in a later section. The Hessian matrix $\nabla^2 F$ can be decomposed into the Gauss-Newton and non Gauss-Newton portions:

$$\nabla^2 F = H_{GN} + H_{NGN}. \tag{3.2.2}$$

The classical Gauss-Newton Hessian shows up for least squares problems, where the loss function is

$$F_{X_k}(w) = \frac{1}{2N_{X_k}} \sum_{i=1}^{N_{X_k}} \|y_i - f(x_i, w)\|^2, \tag{3.2.3}$$

where (x_i, y_i) are data pairs for the input-output map, and $f(\cdot, w)$ is the approximation of the map by the surrogate regressor (the neural network). The decomposition of the Hessian for this problem is

$$\nabla^2 F_{X_k}(w) = \frac{1}{N_{X_k}} \sum_{i=1}^{N_{X_k}} \Bigg(\underbrace{\nabla_w f^T \nabla_w f}_{\text{Gauss-Newton}} - \underbrace{\sum_{j=1}^{d_Y} \nabla_w^2 f_j(x_i, w)(y_i - f(x_i, w))_j}_{\text{non Gauss-Newton}} \Bigg). \tag{3.2.4}$$

The Gauss-Newton Hessian can be generalized to any loss function ℓ that is convex with respect to the surrogate regressor $f(x, w)$ [119]. The generalized formula is

$$H_{GN} = \frac{1}{N_{X_k}} \sum_{i=1}^{N_{X_k}} \nabla_w f(x_i, w)^T \nabla_f^2 \ell(f(x_i, w), y_i) \nabla_w f(x_i, w), \tag{3.2.5}$$

and the non Gauss-Newton portion is the difference between this and the true Hessian [85]. The matrix H_{GN} is always positive definite, and H_{NGN} is indefinite and the source of the negative definite directions of the overall Hessian. Since both matrices are symmetric, they admit spectral decompositions

$$H_{GN} = U_{GN} \Lambda_{GN} U_{GN}^T \tag{3.2.6}$$

$$H_{NGN} = U_{NGN} \Lambda_{NGN} U_{NGN}^T. \tag{3.2.7}$$

The non Gauss-Newton portion can be decomposed into a sum of positive semi-definite and negative semi-definite matrices as follows:

$$H_{NGN} = U_{NGN} \Lambda_{NGN}^{\geq 0} U_{NGN}^T + U_{NGN} \Lambda_{NGN}^{\leq 0} U_{NGN}^T. \tag{3.2.8}$$

A statistical model for the subsampled Hessian can be constructed by summing outer products of vectors from Gaussian distributions that give back the true Hessian in expectation. That is,

$$x_i \sim \rho_+ = \mathcal{N}(0, A), y_i \sim \rho_- = \mathcal{N}(0, B) \tag{3.2.9}$$

$$A = U_{GN}\Lambda_{GN}U_{GN}^T + U_{NGN}\Lambda_{NGN}^{\geq 0}U_{NGN}^T \tag{3.2.10}$$

$$B = -U_{NGN}\Lambda_{NGN}^{\leq 0}U_{NGN}^T \tag{3.2.11}$$

$$H_N = \underbrace{\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T}_{A_N} - \underbrace{\frac{1}{N}\sum_{i=1}^{N} y_i y_i^T}_{B_N} \tag{3.2.12}$$

Note that since $\mathbb{E}_{\rho_+}[x_i x_i^T] = A$ and $\mathbb{E}_{\rho_-}[y_i y_i^T] = B$, by the linearity of expectation and the Monte Carlo error,

$$\lim_{N\to\infty} H_N = \nabla^2 F. \tag{3.2.13}$$

So this model agrees with the true expected risk Hessian asymptotically (in expectation). In this model the number of terms in the sum is analogous to the number of samples used in the expected risk approximation. This model allows us to understand stochasticity of subsampled Rayleigh quotients if they were perfectly Gaussian. The following result characterizes variance in approximation of modes of the Hessian, based on decomposition as a sum of two terms.

Theorem 3.1. *Consider the generalized eigenvalue decomposition of the matrices $A \in \mathbb{R}^{d\times d}$ and $B \in \mathbb{R}^{d\times d}$, that is, the full rank matrices $\Phi \in \mathbb{R}^{d\times d}$ and $\Delta \in \mathbb{R}^{d\times d}$ such that*

$$A\Phi = B\Phi\Delta \tag{3.2.14a}$$

$$\Phi^T A\Phi = diag(a) \tag{3.2.14b}$$

$$\Phi^T B\Phi = diag(b), \tag{3.2.14c}$$

where $a, b \in \mathbb{R}^d$ are vectors that are the result of the diagonalization by the matrix Φ. Note that each column of Φ is unit normal, but the matrix is not

necessarilty orthonormal. If $x_i \sim \mathcal{N}(0, A)$ *and* $y_i \sim \mathcal{N}(0, B)$ *then if*

$$A_N = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \tag{3.2.14d}$$

$$B_N = \frac{1}{N}\sum_{i=1}^{N} y_i y_i^T \tag{3.2.14e}$$

then asymptotically, for large N*, the two Rayleigh quotients obey the following probability distributions*

$$\frac{\phi_k^T[A_N - B_N]\phi_k}{\phi_k^T[A - B]\phi_k} \overset{C.L.T.}{\sim} \mathcal{N}\left(1, \frac{1}{2N}\frac{a_k^2 + b_k^2}{(a_k - b_k)^2}\right) \tag{3.2.14f}$$

$$\frac{\phi_k^T[A_N + B_N]\phi_k}{\phi_k^T[A + B]\phi_k} \overset{C.L.T.}{\sim} \mathcal{N}\left(1, \frac{1}{2N}\frac{a_k^2 + b_k^2}{(a_k + b_k)^2}\right), \tag{3.2.14g}$$

where ϕ_k *is the* k^{th} *column of* Φ*, and* $\overset{C.L.T.}{\sim}$ *means that the distribution of the random variable asymptotically converges the Gaussian distribution by the Central Limit Theorem.*

Proof. For the numerator of both expressions consider the quadratic form $\phi_k^T A_N \phi_k$. Note that x_i obeys unnormalized probability distribution:

$$e^{-\frac{1}{2}x^T A^{-1} x}. \tag{3.2.15}$$

Which can be rewritten as

$$e^{-\frac{1}{2}(\Phi^{-T}\Phi^T x)^T A^{-1}(\Phi^{-T}\Phi^T x)} = e^{-\frac{1}{2}(\Phi^T x)^T(\Phi^{-1}A^{-1}\Phi^{-T})(\Phi^T x_i)}. \tag{3.2.16}$$

Since

$$\Phi^T A \Phi = \mathrm{diag}(a), \tag{3.2.17}$$

this implies that

$$\Phi^{-1}A^{-1}\Phi^{-T} = \mathrm{diag}(a)^{-1}. \tag{3.2.18}$$

So the unnormalized probability distribution can be written as

$$e^{(-\frac{1}{2}(\Phi^T x_i)^T \mathrm{diag}(a)^{-1}(\Phi^T x_i))}, \tag{3.2.19}$$

which implies that the variable $\Phi^T x_i$ obeys Gaussian probability distribution $\mathcal{N}(0, \mathrm{diag}(a))$. It follows that $\phi_k^T x_i$ obeys Gaussian probability distribution $\mathcal{N}(0, a_k)$. One can rewrite the random variable $\phi_k^T x_i$ as

$$\phi_k^T x_i = \sqrt{a_k}\nu_i, \tag{3.2.20}$$

where $\nu_i \sim \mathcal{N}(0,1)$. Then, for the first term in the numerator,

$$\phi_k^T A_N \phi_k = \frac{1}{N}\sum_{i=1}^{N}(\phi_k^T x_i)(\phi_k^T x_i) = \frac{a_k}{N}\sum_{i=1}^{N}\nu_i^2. \tag{3.2.21}$$

Consider $\nu = [\nu_1, \nu_2, \ldots, \nu_N]^T \in \mathbb{R}^N \sim \mathcal{N}(0, I_N)$, then rewrite $\phi_k^T A_N \phi_k$ as a Chi-square variable:

$$\zeta_N = \phi_k^T A_N \phi_k = \frac{a_k}{N}\|\nu\|^2_{\ell^2(\mathbb{R}^d)}. \tag{3.2.22}$$

The Chi-Square variable ζ_N has mean a_k, and variance $2\frac{a_k^2}{N}$. Via an application of the Central Limit Theorem, the Chi-Square variable ζ approximately obeys a Gaussian distribution $\mathcal{N}(a_k, \frac{2a_k^2}{N})$for large N. It follows that

$$\phi_k^T(A_N - B_N)\phi_k \overset{\text{C.L.T.}}{\sim} \mathcal{N}\left(a_k - b_k, \frac{2(a_k^2 + b_k^2)}{N}\right) \tag{3.2.23}$$

$$\phi_k^T(A_N + B_N)\phi_k \overset{\text{C.L.T.}}{\sim} \mathcal{N}\left(a_k + b_k, \frac{2(a_k^2 + b_k^2)}{N}\right), \tag{3.2.24}$$

by a similar construction for B_N and the addition of Gaussians. Further for the relative accuracy quotient

$$\frac{\phi_k^T[A_N - B_N]\phi_k}{\phi_k^T[A - B]\phi_k} \overset{\text{C.L.T.}}{\sim} \mathcal{N}\left(1, \frac{1}{2N}\frac{a_k^2 + b_k^2}{(a_k - b_k)^2}\right) \tag{3.2.25}$$

$$\frac{\phi_k^T[A_N + B_N]\phi_k}{\phi_k^T[A + B]\phi_k} \overset{\text{C.L.T.}}{\sim} \mathcal{N}\left(1, \frac{1}{2N}\frac{a_k^2 + b_k^2}{(a_k + b_k)^2}\right). \tag{3.2.26}$$

□

These fractions represent an finite sum approximation accuracy of the true modes $\phi_k^T(A \pm B)\phi_k$. In nonconvex optimization problems, the Hessian will be indefinite; in any indefinite region there is at least one ϕ_k such that $a_k - b_k < 0$.

If there were no such ϕ_k, then both the the Gauss-Newton and non Gauss-Newton Hessian would be by definition positive semi-definite. The goal of empirical risk minimization for nonconvex problems to find *local minima*, where the Hessian is locally positive semi-definite. Traversing the nonconvex energy landscape in search of local minima means escaping strictly indefinite regions (where there is at least one strictly negative eigenvalues) in search of locally positive semi-definite regions. When one escapes an indefinite region to enter a locally positive semidefinite region this means that for all k, $a_k - b_k$ goes from being strictly negative to being nonnegative. If the Hessian is continuous then by the intermediate value theorem the iterates must pass a point in which $a_k - b_k = 0$. Near these regions in parameter space, the denominator in the variance for the Gaussian approximation approaches zero and the variance can grow unbounded; information about these directions is likely to be dominated by noise.

In general if iterates pass through a point in parameter space where the true empirical risk Hessian has an eigenvalue that changes sign (a local inflection point), the variance in approximation of eigenvalues in these directions can grow unbounded. If the Hessian does not change too rapidly, perturbations in directions with large eigenvalues are less likely to be near a local inflection point, so larger modes of stochastic Hessians may be easier to resolve, i.e. have less statistical noise.

Overfitting occurs when a stochastic optimizer starts tuning parameters to noisy information specific to the data used during training that is not representative of the true underlying mapping. The goal of empirical minimization in practice is to find a highly generalizable candidate local minima, that is, one that avoids overfitting. If possible, one should avoid incorporating noisy information in search directions. When the stochastic Hessian is indefinite, it may exhibit potentially unbounded variance if eigenvalues change sign. Information in such directions can be dominated by noise, and one should avoid using noise dominated information, if possible.

This relationship between noise and indefiniteness is observed in numerical experiments presented later in this chapter. When the stochastic Hessian has both negative and positive eigenvalues the stochastic Rayleigh quotients taken with respect to many mini batches exhibit very high sample variance. When the Hessian is positive definite the variance is much smaller.

In what follows, I analyze the extension of inexact Newton methods to the fully stochastic nonconvex regime.

3.3 Inexact Newton Methods

A subsampled inexact Newton method as described in [42] is a method for which the Newton system (3.1.4) is solved inexactly, and the linear solve is terminated when the following condition is satisfied:

$$\|\nabla^2 \overline{F}_{S_k} p_k + \nabla \overline{F}_{X_k}\| \leq \eta_k \|\nabla \overline{F}_{X_k}\|. \tag{3.3.1}$$

When the gradient is large, the tolerance for inexactness is high. The tolerance tightens as one nears the solution. This avoids unnecessary work in the linear solves far from the solution, but still retains super-linear or quadratic convergence near the solution. Optimal choices of η_k are discussed in the papers of Eisenstat and Walker [42, 43]. I establish the following local convergence rate for a stochastic inexact Newton method for the choice of $\eta_k \leq \|\nabla \overline{F}_{X_k}\|$.

Theorem 3.2 (Local convergence for stochastic inexact Newton methods with gradient norm forcing)**.** *Let w^* be a stationary point and suppose that assumptions A1-A4 hold, let*

$$\mu = \min\left\{\|\nabla^2 F(w^*) + \gamma I\|^{-1}, \|[\nabla^2 F(w^*) + \gamma I]^{-1}\|\right\}, \tag{3.3.2}$$

and assume that

(a) $w_k \in B_\delta(w^*)$ with $\delta < \frac{2\mu}{L_{N_{S_k}}}$,

(b) $-\epsilon_H I \preceq \nabla^2 F_{S_k}$ for all S_k and for all $w \in B_\delta(w^*)$,

(c) The Tikhonov regularization parameter is chosen such that $\gamma > \epsilon_H$,

(d) $\|\nabla^2 \overline{F}_{S_k}(w_k)p_k - \nabla \overline{F}_{X_k}(w_k)\| \leq \eta_k \|\nabla \overline{F}_{X_k}(w_k)\|$ *with* $\eta_k \leq \|\nabla \overline{F}_{X_k}(w_k)\|$.

Then for the iterate $w_{k+1} = w_k + \alpha_k p_k$, *one has the following bound:*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2, \tag{3.3.3}$$

where

$$c_0 = \frac{1}{\gamma - \epsilon_H}\left[\frac{\alpha_k v}{\sqrt{N_{X_k}}}\left(1 + \frac{v}{\sqrt{N_{X_k}}}\right)\right] \tag{3.3.4a}$$

$$c_1 = \frac{1}{\gamma - \epsilon_H}\left(L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} + \frac{2\alpha_k v\mu}{\sqrt{N_{X_k}}}\right) \tag{3.3.4b}$$

$$c_2 = \frac{1}{\gamma - \epsilon_H}\left(\frac{M}{2} + \alpha_k \mu^2\right). \tag{3.3.4c}$$

For proof see Section 3.8. Assumption (a) states that w_k is sufficiently close to an optimum. Assumption (b) states that eigenvalues of the Hessian are not too negative, and assumption (c) guarantees the Tikhonov regularized Hessian is invertible. Assumption (d) is the Eisenstat-Walker forcing condition. Ideally the constants c_0, c_1, and c_2 will be as small as possible. The constant c_0 will be small when the Monte Carlo approximation of the gradient is good. The constant c_1 will be small when the Monte Carlo approximation of the gradient and Hessian are both good, and the full Newton step $\alpha_k = 1$ can be taken. The constant c_2 will be small when the Hessian is well conditioned.

This theorem does not address how expensive the method may be per iteration. The per iteration cost of the method will depend on the computational method for approximating the Hessian, which depends on the spectral properties of the Hessian and the batch size. In the next subsections, I will analyze how solving the Newton system approximately using Krylov methods affects the convergence rate.

3.4 Inexact Newton-Krylov Methods

Krylov methods are the preferred linear solver for inexact Newton methods. In this section I consider their extension to stochastic nonconvex problems.

Definition 3.1 (Krylov Subspace)**.** *Given $A : \mathbb{R}^d \to \mathbb{R}^d$ and $y \in \mathbb{R}^d$, define the m^{th} Krylov subspace as the linear subspace $\mathcal{K}_m(A, y) \subset \mathbb{R}^d$ as*

$$\mathcal{K}_m(A, y) = span\{y, Ay, \ldots, A^{m-1}y\}. \tag{3.4.1}$$

Given $p_0 = 0$ as an initial guess, stochastic Newton-Krylov methods approximate

$$p = -[\nabla^2 \overline{F}_{S_k}]^{-1} \nabla \overline{F}_{X_k} \approx p_m \in \mathcal{K}_m(\nabla^2 \overline{F}_{S_k}, -\nabla \overline{F}_{X_k}) \tag{3.4.2}$$

via a Galerkin projection. Krylov methods require only the action of a matrix on vectors; access to the entries of the matrix is not required. In this work the Krylov methods considered are conjugate gradients (CG), the minimal residual method (MINRES), and the generalized minimal residual method (GMRES). GMRES applies to indefinite matrices, MINRES applies to symmetric indefinite matrices. CG can be adapted to symmetric indefinite matrices by a simple modification. A generic stochastic inexact Newton-Krylov method is described below.

Algorithm 1: Inexact Newton-Krylov Methods

```
Given w_0
while not converged do
    if ||∇F̄_{X_k}|| ≤ ε_g and λ_min(∇²F_{S_k}) ≥ −ε_H then
        break
    end
    Given ||∇F̄_{X_k}|| compute η_k via Eisenstat-Walker
    Solve ||∇²F̄_{S_k} p_k + ∇F̄_{X_k}|| ≤ η_k ||∇F̄_{X_k}|| via a Krylov method
    α_k given or computed via line search
    w_{k+1} = w_k + α_k p_k
end
```

3.4.1 Local convergence rates

In the case that CG is used for the linear solve, Bollapragada et al. have derived a local convergence rate for the *semi-stochastic* case, in which the gradient is not subsampled [15]. I extend this analysis to the fully stochastic setting, including the dependence of the convergence constants on the parameters α_k and γ.

Theorem 3.3 (Local convergence of stochastic inexact Newton CG (INCG), extension of Lemma 3.1 of [15])**.** *Let w^* be a stationary point, suppose assumptions A1-A4 hold, and the iterates $\{w_k\}$ are generated by the stochastic inexact Newton CG method, the direction p_k^r is found in $r \ll d$ steps (for justification see section 3.4.2), and there exists $\epsilon_H > 0$ such that $-\epsilon_H I \preceq \nabla^2 F_{S_k}(w_k)$ and $\gamma > \epsilon_H$. Then,*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2 \tag{3.4.3}$$

where

$$c_0 = \frac{\alpha_k v}{(\gamma - \epsilon_H)\sqrt{N_{X_k}}} \tag{3.4.4a}$$

$$c_1 = \frac{1}{\gamma - \epsilon_H}\left[L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} + 2\alpha_k L_{N_{S_k}}\sqrt{\kappa_{N_{S_k}}}\left(\frac{\sqrt{\kappa_{N_{S_k}}} - 1}{\sqrt{\kappa_{N_{S_k}}} + 1}\right)^r\right] \tag{3.4.4b}$$

$$c_2 = \frac{M}{2(\gamma - \epsilon_H)}, \tag{3.4.4c}$$

and $\kappa_{N_{S_k}}$ is the condition number of the Tikhonov regularized Hessian.

A proof of this result can be found in Section 3.8. For fast convergence, one wants the constants c_0, c_1 and c_2 to be as small as possible. The constant c_0 is small when the Monte Carlo approximation error for the gradient is small. The term c_1 will be small when the full Newton step ($\alpha_k = 1$) can be taken, when the Monte Carlo approximation of the Hessian is accurate, and the linear solve error is small after r steps of CG. The constant c_2 will be small when the Hessian is well conditioned.

Remark. *In both of the theorems up to this point (Theorem 3.2 and Theorem 3.3), the constant c_0 depends only on the Monte Carlo error in the gradient calculation. In the semi-stochastic setting (where only the Hessian is subsampled) each of these convergence rates is then linear-quadratic.*

Worst case bounds for Krylov method convergence can often be established based on the condition number of the matrix. The convergence of Krylov methods will more generally depend on the entire spectrum of the Hessian, and will benefit from spectral clustering, this is explored in the next section. Preconditioners can be used to enhance convergence by reducing the condition number of the matrix, or clustering the spectrum. In the case of MINRES, worst case bounds can be established based on the condition number. GMRES achieves superior convergence if the spectrum of H resides in an interval that does not include the origin. I present a bound for the convergence rate of stochastic inexact Newton GMRES and MINRES algorithms in Section 3.8.

3.4.2 Superior Approximation for Clustered Eigenvalues

Krylov subspaces are intimately related to spaces of polynomials. The Krylov subspace $\mathcal{K}_m(A, y)$ is the space of all vectors $x \in \mathbb{R}^d$ that can be written as $x = p(A)y$ where $p \in \mathbb{P}_{m-1}$, the space of all polynomials of degree $m-1$ or less. If the generating vector y is not degenerate—in the basis of eigenvectors of A, none of its components are zero—then there is a natural isomorphism between $\mathbb{P}_{m-1}$ and $\mathcal{K}_m$ defined by [114]:

$$\mathbb{P}_{m-1} \ni q \mapsto x = q(A)y \in \mathcal{K}_m(A, y). \tag{3.4.5}$$

This isomorphism with polynomials allows us to analyze Krylov solution by instead analyzing polynomials. Let $A = U\Lambda U^T$ denote the eigenvalue decomposition of A, with $\Lambda = \text{diag}(\lambda_k)$ and u_k is the k^{th} column of U. Let $\mathcal{Q}_m$ denote the set of all m^{th} order polynomials with constant term 1, that is

$$\mathcal{Q}_m = \{q \in \mathbb{P}_m | q(0) = 1\}. \tag{3.4.6}$$

I review the following well-known results for CG, GMRES and MINRES.

Theorem 3.4 (Krylov methods and minimum polynomials)**.** *Let x_m be the m^{th} CG iterate for solving $Ax = b$, and $x^* = A^{-1}b$. The following relationship holds:*

$$\|x^* - x_m\|_A^2 = \min_{q \in \mathcal{Q}_m} \sum_{k=1}^{n} \lambda_k q(\lambda_k)^2 (u_k^T e_0)^2 \tag{3.4.7}$$

where $e_0 = x^* - x_0$. Let x_m be the m^{th} GMRES (with no restarts) or MINRES iterate for solving $Ax = b$. Then the following relationship holds:

$$\|b - Ax_m\|^2 = \min_{q \in \mathcal{Q}_m} \sum_{k=1}^{n} q(\lambda_k)^2 (u_k^T r_0)^2 \tag{3.4.8}$$

where $r_0 = b - Ax_0$ is the initial residual.

These results are canonical; see Appendix section 3.1 for proof sketches. When eigenvalues are clustered, a lower degree polynomial is better able to minimize either (3.4.7) or (3.4.8). This means that CG, GMRES, and MINRES will perform better than low rank approximations when the eigenvalues are clustered. Due to the additional λ_k in equation (3.4.7), CG will eliminate errors in the subspaces corresponding to large eigenvalues more aggressively than in the subspaces corresponding to small eigenvalues (GMRES and MINRES do not have this property).

An issue arises however when the Hessians are *stochastic*. Larger eigenvalues represent directions with more information content, and smaller eigenvalues can be dominated by noise [7]. This means that Krylov methods may pay too much attention to modes of the Hessian that are dominated by noise, and can lead to Newton directions that overfit to the noise in the specific instance of the stochastic Hessian at that iteration.

This poses a serious issue for Krylov methods; they pay attention to the whole spectrum of the Hessian, but it is the dominant modes of the Hessian that matter the most. For this reason CG is more desirable than GMRES and MINRES in this context since it emphasizes larger eigenvalues of the

Hessian, which have more signal. Inexact Newton-Krylov methods can be made somewhat robust to saddle points. At a given Krylov iteration with a search direction update v_m, an un-normalized Rayleigh quotient $v_m^T H v_m$ can be calculated (in the case of CG this term is already calculated). If this quantity is negative, then the vector v_m points in a direction that is dominated by negative curvature, noting that the direction can also have components in eigenvectors of the Hessian corresponding to positive curvature. One can then terminate the Krylov solve early, without modifying the search direction ${p_m}^3$.

While early termination procedures can be employed to avoid taking steps towards saddle points, the only thing that can be done to avoid incorporating these noise dominated modes into the expansion of the solution in the Krylov space is to dominate the spectrum with regularization that shifts the spectrum up, and eventually pollutes the second order information and gradient. In what follows I propose a novel method (low rank saddle free Newton), which avoids this issue by using a low rank approximation of the Hessian that avoids these noise dominated modes.

3.5 Low Rank Saddle Free Newton

Let the spectral decomposition of the Hessian be given as follows:

$$\nabla^2 F = U \Lambda U^T = \sum_{i=1}^{d_W} \lambda_i u_i u_i^T, \tag{3.5.1}$$

where the eigenvalues λ_i are sorted such that $|\lambda_i| \geq |\lambda_j|$ for all $i > j$, and $u_i \in \mathbb{R}^{d_W}$ are the corresponding eigenvectors. If the Hessian is invertible without regularization, exact Newton rescales the negative gradient component-wise in the Hessian eigen-basis, by the corresponding eigenvalue,

$$p = -[\nabla^2 F]^{-1} \nabla F = -\sum_{i=1}^{d} \frac{1}{\lambda_i} (\nabla F^T u_i) u_i. \tag{3.5.2}$$

[3]Note that for the system $Ax-b$, CG is a minimizer of the quadratic $\frac{1}{2}x^T Ax - x^T b$, which is only bounded below for positive semi-definite A, one must terminate when negative curvature directions are encountered to avoid spurious unbounded solutions.

When an eigenvalue is negative, the components of the gradient in this direction will change sign and point towards the saddle point, instead of away. Therefore exact Newton iterates may converge to saddle points.

Beyond early termination, Newton Krylov methods cannot facilitate fast escape from indefinite regions. If one has access to the spectral decomposition of the Hessian, fast escape from indefinite regions can be achieved by flipping the negative eigenvalues of the Hessian to be positive [56], this is known as the saddle free Newton (SFN) algorithm in the machine learning community [41]. In the SFN method, one solves $|\nabla^2 F|p = -\nabla F$, where $|\nabla^2 F| = U|\Lambda|U^T$, I refer to this as the absolute value of the Hessian. The key to the method is scalability; one wants to approximate the absolute value of the Hessian inexpensively (i.e. in a matrix-free way). Naïve formation of the Hessian is $O(d_W^2)$ work and factorizing this matrix is $O(d_W^3)$. Ideally one wants to find an approximation of the absolute value of the Hessian for less than $O(d_W^2)$ work.

In the SFN method of Dauphin et al [40], the absolute value of the Hessian is approximated using a Krylov method (stochastic Lanczos quadrature). In their method, the absolute value of the Hessian is approximated in the subspace spanned by the first k-Lanczos vectors of the empirical risk Hessian. They employ a trust region framework that is equivalent to Levenberg-Marquardt damping (see for example [95]), which allows for an efficient inversion of the Hessian. The method requires k Hessian matrix vector products, and in many problems k may be taken much smaller than d_W.

I propose a different saddle free Newton algorithm that uses low-rank instead of Krylov to approximate the absolute value of the Hessian. Recent empirical studies show that neural network training Hessians are typically numerically low rank, and when away from local minima often have at least one large magnitude negative eigenvalue [3, 55, 117]. If the Hessian is low rank, the system

$$(|H| + \gamma I)p = -g \tag{3.5.3}$$

can be efficiently inverted using the Sherman Morrison Woodbury formula. The γI term in Equation (3.5.3) shows up when Tikhonov regularization [128] or Levenberg-Marquardt damping are used [92]. I term this method the low rank Saddle Free Newton method (LRSFN). There are benefits to this algorithm over the original SFN and inexact Newton Krylov methods for stochastic nonconvex optimization. First low rank decompositions only attempt to resolve dominant modes of the Hessian, Krylov methods attempt to resolve the entire spectrum of the Hessian, which may include noisy directions that can lead to poor generalization properties. Second, low rank factorizations can be efficiently approximated using randomized methods, which can be easily parallelized, unlike classical Krylov methods, which are inherently sequential processes. Finally, in the vicinity of a local minimum, LRSFN can obtain fast local convergence.

3.5.1 Low Rank vs. Krylov

There are benefits to using randomized algorithms to approximate the low rank eigenvalue decomposition over Krylov methods, such as the stochastic Lanczos quadrature used in [41]. On one hand, the Eckart-Young Theorem states that the low rank approximation is optimal in the spectral and Frobenius norms; it gives the following bound for the Hessian approximation error in the both norms:

$$\|H - H_r\|_2 = \lambda_{r+1}, \tag{3.5.4}$$

$$\|H - H_r\|_F = \sqrt{\sum_{i=r+1}^{d} \lambda_i^2}. \tag{3.5.5}$$

On the other hand, Krylov approximations such as Lanczos are optimal in the sense of certain polynomial approximations; for more details see Section 3.4.2, [114].

I argue that randomized low rank approximation is better in this setting. It (a) leads to solutions that generalize better when the Hessian is subsampled, and (b) is better at escaping saddle points.

(a) The objective function is most sensitive to perturbations of w in directions corresponding to eigenvalues of large magnitude, since the energy landscape has large curvature in these directions. These directions typically persist when different sets of subsamples are used to approximate the Hessian. Directions corresponding to eigenvalues of small magnitude are less important since the objective function is less sensitive to perturbations in these directions. When the Hessian is indefinite, these directions may exhibit very high variance, as is suggested by analysis in Section 3.2, and demonstrated empirically in numerical results. These directions are likely to change drastically for different subsamples of the data, and one should avoid using information from these modes, as it may lead to overfitting.

Since Krylov methods approximate the whole spectrum, they waste computational effort attempting to approximate eigenvalues of small magnitude that vary depending on the random subsamples used. Low rank approximation only approximates the large eigenvalues, and therefore leads to solutions that generalize better.

(b) Krylov subspace approximations are heavily dependent on the initial vector for the subspace. In Newton-Krylov methods such as SFN, the gradient is the initial vector. However in the vicinity of a saddle point the gradient may have small components in eigenvector directions corresponding to eigenvalues that are negative but large in magnitude. Randomized low rank methods are better than Krylov methods at capturing these large magnitude directions when the gradient is small in these directions. Hence LRSFN pushes iterates away from saddle points more strongly than Krylov-based Saddle Free Newton.

3.5.2 Scalable approximation and efficient inversion using randomized methods

Low rank factorizations can be efficiently approximated using randomized methods. Randomized methods are likely to span the range space of the operator with high probability (see [58, 89]), and can leverage concurrency

which is not available to inherently sequential Krylov methods. This leads to scalable ways to efficiently approximate Hessians for very large dimensional problems.

Neural network training Hessians are often observed to have fast spectral decay[3, 55, 117] making this a reasonable approximation in many settings. However even if they are not low rank, the rank of the approximation can be tuned to a particular noise level in the spectrum to avoid incorporating noisy modes. This makes a low rank approximation potentially useful even in a regime where the Hessian is not numerically low rank. In these methods a low rank approximation of the matrix $\nabla^2 F$ is approximated via the operator's action on random vectors (drawn from a distribution ρ) that span the dominant modes of the operator with high probability. This leads to approximation errors in expected value with respect to ρ that scale with the values of the truncated eigenvalues.

Classical Krylov processes are inherently serial since they are based on power iterations of the matrix being used; the matrix-vector products used in Krylov methods depend on previous computations. The matrix-vector products required by randomized low rank methods are independent and therefore easily parallelized. This means that randomized low rank methods can leverage concurrency that Krylov methods cannot. The key computation in the randomized low rank method is a matrix-matrix product:

$$\nabla^2 F(w)\Omega, \tag{3.5.6}$$

where $\Omega \in \mathbb{R}^{d_W \times r+p}$ is the Gaussian random matrix drawn from the probability distribution ρ, r is the target rank, and p is the oversampling parameter used to improve the approximation. The matrix Ω can be block-column partitioned as

$$\Omega = \left[\Omega_0 | \Omega_1 | \cdots | \Omega_{N_{\text{proc}}}\right] \tag{3.5.7}$$

and broadcasted to N_{proc} independent parallel processes. This way the computation of the Hessian action will only take as long as the process with with

most columns in the block partition. This parallelism can leverage modern computing infrastructure to make the method scalable to problems that have higher rank. This computational concurrency presents a serious advantage over inherently serial Krylov procedures, making randomized low rank Newton methods the clear choice for massive scale optimization problems.

3.5.3 Low Rank Saddle Free Newton Algorithm

I use low rank approximation obtained via randomized methods, in combination with the Sherman-Morrison-Woodbury formula, to solve the modified Newton system (**??**) as follows:

$$p_k = -\left[\frac{1}{\gamma} I_d - \frac{1}{\gamma^2} U_r \left(|\Lambda_r|^{-1} + \frac{1}{\gamma} I_r \right)^{-1} U_r^T \right] g_k. \tag{3.5.8}$$

The low rank saddle free Newton (LRSFN) method is summarized in Algorithm 2.

Algorithm 2: Randomized Low Rank Saddle Free Newton

Given w_0
while *not converged* **do**
  if $\|\nabla \overline{F}_{X_k}\| \leq \epsilon_g$ *and* $\lambda_{min}(\nabla^2 F_{S_k}) \geq -\epsilon_H$ **then**
    break
  end
  Calculate $U_r^{(k)} \Lambda_r U_r^{(k)T} \approx H^{(r)}$ via randomized methods[58]
  Calculate $p_k = -\left[\frac{1}{\gamma} I_d - \frac{1}{\gamma^2} U_r^{(k)} \left(|\Lambda_r^{(k)}|^{-1} + \frac{1}{\gamma} I_r \right)^{-1} U_r^{(k)T} \right] \nabla \overline{F}_{X_k}$
  α_k given or computed via globalization
  $w_{k+1} = w_k + \alpha_k p_k$
end

When globalization is used (line search or trust region) there are very few hyperparameters that require tuning in the method. This presents a significant advantage over first order methods, where ideal step lengths are problem specific and typically scale with the inverse of the Lipschitz constant for the Hessian. The main hyperparameter to be decided is the rank used in the Hessian approximation. Adaptive range finding (ARF) methods are used to iteratively construct a low rank approximation of the full matrix and increase the rank until an approximation error is satisfied (see for example

Algorithm 4.2 in [58]). When approximating a matrix A, via a low rank approximation A_r, one wants to find r such that for a given tolerance $\epsilon > 0$ the convergence criteria is satisfied:

$$\|A - A_r\| \leq \epsilon, \tag{3.5.9}$$

where the norm is usually taken to be the spectral (ℓ^2) norm. This approximation error too can be approximated efficiently using randomized methods. Since Gaussian random vectors span all of parameter space with high probability, one can sample $\xi \sim \rho$, and substitute the condition

$$\|A\xi - A_r\xi\| \leq \epsilon\|\xi\|, \tag{3.5.10}$$

which agrees with the true error tolerance with high probability. In typical applications one only cares about approximating the matrix to a desired precision, and the standard adaptive range finding method is sufficient for this aim. In stochastic nonconvex optimization, it is useful to approximate to high accuracy *only in Hessian eigenmodes with low variance*. I suggest a procedure that modifies the existing ARF procedure to truncate the decomposition before fitting modes with high variance. If the low rank eigenvalue decomposition of the subsampled matrix A_X is $A_r = U_r\Lambda_r U_r^T$, then one can compute samples of Rayleigh quotients of the last few eigenvectors ($j \in \{r_k, \cdots r\}$ for small k) with respect to $\{X_i\}_{i=1}^{N_{\text{samples}}}$ subsamples of X:

$$\{u_j^T A_{X_i} u_j\}_{i=1}^{N_{\text{samples}}}. \tag{3.5.11}$$

When the sample average variance of the Rayleigh quotients becomes larger than a predetermined noise tolerance, once stops the adaptive range finding procedure. This modified procedure terminates when the first of these two conditions is met. I name this procedure noise-aware adaptive range finding (NAARF).

In practice sampling the Rayleigh quotients can be computationally prohibitive. The process can be made cheaper by taking k or N_{samples} smaller.

Experimental observations suggest that the noise level and the rank of the Hessian do not change drastically other than the first few steps after an initial guess or when a method enters a positive semidefinite region. For this reason, I recommend using this procedure sparingly, and then fixing the rank for a certain number of iterations before recalculating the rank. Near initial guesses, not much Hessian information is necessary to compute a quality descent direction, so a small rank can be used for the first few iterations to introduce further computational economy as in Krylov tolerances based on the Eisenstat-Walker conditions (see Section 3.3).

3.5.4 Local convergence for the stochastic low rank Newton method

When the Hessians are low rank (which is experimentally observed in the vicinity of local minima, see numerical results), LRSFN can obtain fast local convergence.

Convergence to a local minima is only possible in regions where the Hessian is positive semi-definite. In this case the eigenvalues of the Hessian are all positive initially and the LRSFN method is the same as a general low rank Newton method. I have the following bound that establishes the conditions for fast convergence in the vicinity of a local minimum.

Suppose that for each w_k and S_k, one has the truncated eigenvalue decomposition $H_r^{(k)} = [\nabla^2 F_{S_k}]_r = U_r^{(k)} \Lambda_r U_r^{(k)T}$ for the empirical risk function, and the iterates

$$w_{k+1} = w_k - \alpha_k [H_k^{(r)} + \gamma I]^{-1} \nabla \overline{F}_{X_k}(w). \tag{3.5.12}$$

Theorem 3.5 (Local convergence of stochastic low rank Newton)**.** *Let* $\{w_k\}$ *be the iterates generated by* (3.5.12)*, let* w^* *be a stationary point and suppose that assumptions A1 - A4 hold, then for each* k

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1 \|w_k - w^*\| + c_2 \|w_k - w^*\|^2, \tag{3.5.13}$$

where

$$c_0 = \frac{\alpha_k v}{|\overline{\lambda_r^{(k)}} + \gamma|\sqrt{N_{X_k}}}, \tag{3.5.14a}$$

$$c_1 = \frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\left[L_{N_{S_k}}|1-\alpha_k| + \mathcal{E}|\overline{\lambda_{r+1}^{(k)}}| + \gamma + \frac{\sigma}{\sqrt{N_{S_k}}}\right], \tag{3.5.14b}$$

$$c_2 = \frac{M}{2|\overline{\lambda_r^{(k)}} + \gamma|}. \tag{3.5.14c}$$

Here I define $\overline{\lambda_r^{(k)}} = \mathbb{E}_k[\lambda_r^{(k)}]$. *The error coefficient* $\mathcal{E} = 1$ *when the truncated low rank spectral decomposition is exact, and* $\mathcal{E} = \left(1+4\frac{\sqrt{d(r+p)}}{p-1}\right)$ *in the case that it is calculated using randomized SVD.*

A proof of this result can be found in Section 3.8.

For fast convergence, one wants the constants c_0, c_1 and c_2 to be as small as possible. The constant c_0 is small when the Monte Carlo approximation error for the gradient is small. The constant c_1 has errors from step length (if $\alpha \neq 1$), the Hessian Monte Carlo approximation, the low rank Hessian approximation, and in the case of randomized SVD, the additional approximation factor $\mathcal{E}$. When the Hessian has low rank, the approximation error by low rank factorization will be small. The Hessian is often low rank in machine learning applications [3, 55, 117]. I also observe that the Hessian has low rank in our numerical experiments. The constant c_2 will be small when the Hessian is well conditioned.

3.5.5 Comparing costs: gradient vs. Hessian

So far the development has been based on the finite sum optimization problem (3.0.2). I restrict the discussion at this point to neural network training. The dominant costs associated with neural network training are the evaluations of the neural network and its derivatives. The gradient can be formed efficiently using an adjoint process (referred to as back propagation in the neural network literature), which amounts to a forward and backward evaluation of the neural network [113]. The action of the Hessian on a vector

can be formed using an adjoint based method, by an additional forward and backward evaluation of the neural network as described by Pearlmutter [104]. I refer to the pair of one forward and one backward evaluation of the neural network as a *sweep*.[4]

For a given iteration of a low rank Newton method with rank r and oversampling parameter p, the number of network sweeps used to construct the low rank Hessian approximation can be expressed as follows:

$$\#(\text{Low rank Hessian sweeps}) = 2C(r+p)N_{S_k}. \tag{3.5.15}$$

Here $C = 1, 2$ depending on if single pass or double pass algorithms are used for randomized SVD [58]. I use the double pass algorithm. The total neural network sweeps for the double pass version of the low rank SFN algorithm, including the cost of computing the gradient, is then

$$\#(\text{Low rank Newton sweeps}) = \left(N_{X_k} + 4(r+p)N_{S_k}\right). \tag{3.5.16}$$

The cost of the associated linear algebra for randomized SVD will yield an additional $O(dr^2 + r^3)$ operations. For the inexact Newton-Krylov method with r Krylov iterations,

$$\#(\text{Inexact Newton-Krylov sweeps}) = \left(N_{X_k} + 2rN_{S_k}\right). \tag{3.5.17}$$

Previous analysis (Theorem 3.2, Theorem 3.5, and Theorem 3.3, Theorem 3.6) suggests that taking N_{X_k} large is important if one desires fast convergence. As for N_{S_k}, Bollapragada et al. use convergence rates similar to the ones presented in previous analysis to derive conditions on how to increase N_{S_k} to maintain superlinear convergence rates [15]. Since the computational cost will grow with this increase in batch size, I take $N_{S_k} \ll N_{X_k}$ fixed.

[4]Note that the forward evaluation for the gradient will typically be nonlinear, while the backward evaluation for the gradient and the forward and backward evaluations for the Hessian-vector product will be affine since they involve the transpose of the Jacobian of the forward mapping in the case of the gradient, and similar terms for the Hessian. I count these sweeps all the same, even though the Hessian sweeps may be cheaper as is the case in inverse problems and PDE constrained optimization.

In the next section, I show numerically that the subsampled Hessian still provides a good approximation of the true Hessian, even when N_{S_k} is small relative to N_{X_k}. This empirical observation in combination with (3.5.16) and (3.5.17) suggests that the per iteration cost of a stochastic Newton method is not substantially more than the per iteration cost of gradient descent. But stochastic Newton methods will have superior convergence properties.

3.6 Numerical Experiments

I demonstrate the effectiveness of the INCG and LRSFN algorithms on three classes of problems. I investigate standard benchmark problems in machine learning: classification and autoencoder problems on the MNIST and CIFAR10 datasets respectively [72, 74]. I then investigate regression problems arising in parametric map neural network construction.

My focus is on comparing how the optimization methods perform for fixed neural network training problems, in a fixed number of neural network sweeps (I define a *sweep* as a forward or adjoint evaluation of the network). As was stated prior, Hessian vector products require only an additional forward and backward pass of the neural network, which allow us to compare computational costs between first and second order methods in a straightforward way.

Finding optimal architectures for a given input-output representation is outside of the scope of this work. I use neural network architectures that are inspired by those used in practical settings, but with smaller configuration spaces to reduce computational complexity.

I compare the performance of the LRSFN against an existing implementation of the SFN algorithm [46], as well as Adam gradient descent (GD), and stochastic gradient descent (GD). In my implementation of LRSFN we implement different batching for the Hessian and gradient. In the SFN code of [46] this is not implemented, so I used the same data for the gradient and Hessian in this method. I make some direct comparisons between the

methods, where both use the same gradient and Hessian data. I compare all methods based on the number of neural network sweeps, as this is the primary computational cost in neural network training. The goal of the empirical risk minimization is to find a weight configuration $w^* \in \mathbb{R}^{d_W}$ that performs the best on unseen testing data, i.e. good generalization. During training I evaluate the performance of the networks both on data used to train the network, as well as data set aside for testing.

I investigate the stochasticity of subsampled Hessian Rayleigh quotients to supplement analysis in Section 3.2. Experimental observations of the Hessian agree with other works that observe spectral collapse and indefiniteness of the Hessian far away from local minima [3, 55, 117]. In addition the experimental observations agree with the theoretical predictions from Section 3.2 that the stochastic Rayleigh quotients exhibit high variance in indefinite regions, but significantly less variance near local minima.

3.6.1 MNIST Classification

For MNIST classification I use a simple convolutional residual neural network structure, such as those in [61]. For a convolutional resnet with depth 2 the size of the configuration space is $d_W = 8,460$. Note that this architecture was chosen to be simple enough that I could run many tests on it. It is however not typical of what is used in image classification and for this reason the trained neural network gets classification accuracies less than state of the art architectures.

Of the $70,000$ MNIST image label data, I use $60,000$ for training and $10,000$ for testing. The output of the neural network for a given image $f(x_i, w)$ is the prediction of the class (integer between 0 and 9) for the image. For the first set of examples I do not normalize the MNIST images, this effects the scaling of the problem, and exposes some issues for convergence. The classification problem uses a cross-entropy loss function of the form:

$$F_{X_k}(w) = -\frac{1}{N_{X_k}} \sum_{i=1}^{N_{X_k}} y_i \log(f(x_i, w)) \tag{3.6.1}$$

Analysis in previous sections suggests that convergence rates are most effected by the gradient subsampling errors, for this reason I first use larger batch sizes. For this I compare each method for $N_{S_k} = N_{X_k} = 10,000$ and $1,000$. I compare INCG, LRSFN and GD all using line search against Adam and SGD using fixed steps of $\alpha_k = 0.01$, and SFN which implements trust region for globalization. For INCG and LRSFN I use $N_{S_k} = 1,000$ when $N_{X_k} = 10,000$ and $N_{S_k} = 100$ when $N_{X_k} = 1,000$. As was previously noted, the implementation of SFN that I compare against does not implement variable Hessian batching, so I in this case I use $N_{X_k} = N_{S_k}$. I make direct comparisons of all second order methods with $N_{X_k} = N_{S_k}$ later. Results can be seen in the Table 3.1 below.

	Train accuracy	Test accuracy
LRSFN LS $N_{X_k} = 1,000$, $r = 20$	99.1	90.2
INCG LS $N_{X_k} = 1,000$	100.0	88.3
SGD $\alpha_k = 0.01$, $N_{X_k} = 1,000$	100.0	87.9
GD LS $N_{X_k} = 1,000$	100.0	86.7
LRSFN LS $N_{X_k} = 10,000$, $r = 20$	92.6	86.1
Adam $\alpha_k = 0.01$, $N_{X_k} = 1,000$	100.0	85.6
SGD $\alpha_k = 0.01$, $N_{X_k} = 10,000$	100.0	76.7
INCG LS $N_{X_k} = 10,000$	99.6	75.4
GD LS $N_{X_k} = 10,000$	100.0	74.0
SFN $N_{X_k} = 10,000$	65.9	63.6
Adam $\alpha_k = 0.01$, $N_{X_k} = 10,000$	100.0	60.3
SFN $N_{X_k} = 10,000$	49.1	50.1

Table 3.1: Summary of results for $N_{X_k} = 10,000$ and $1,000$

While analysis suggested that convergence rates are most effected by gradient subsampling error, and convergence should improve when taking large batch sizes, this is only true asymptotically (in the vicinity of an optimizer). This result is also for convergence rate per optimization iteration, and says nothing about convergence with respect to fixed computational work.

Results show that the optimizers perform better for the classification problem with smaller batch size. This is a well known empirical observation

in machine learning, and has led to the popularization of small batch size stochastic approximation (SA) methods. For this reason I focus on the case $N_{X_k} = 1000$, with varying N_{S_k} in order to focus on the effect of the Hessian subsampling error alone, for fixed gradient subsampling.

For this set of numerical tests with fixed gradient batch size, I split the results into two main categories: methods using globalization (INCG and LRSFN using line search, and SFN which implements trust region), and those that use a fixed step size throughout the training. The former category have an advantage in that no hyperparameter tuning is required. The latter category have an advantage in that they may perform better; although costly hyperparameter tuning may be required to realize optimal performance.

Comparison of methods using globalization

I start with the globalization methods, where I study the effect of Hessian batch size. I compare INCG LS and LRSFN LS with GD LS and SFN TR. For INCG and LRSFN I test several different Hessian batch sizes, and for LRSFN I study different fixed ranks $r = 20, 40$. For INCG and LRSFN I use 8 and 12 backtracking iterations in the line search respectively. The results for the methods are summarized below. Accuracy here is the percentage of correct predictions.

	Train accuracy	Test accuracy
LRSFN $N_{S_k} = 100$, $r = 30$	95.5	90.5
LRSFN$N_{S_k} = 100$, $r = 20$	99.1	90.2
LRSFN $N_{S_k} = 200$, $r = 30$	92.8	90.2
INCG $N_{S_k} = 200$	100.0	89.91
LRSFN $N_{S_k} = 200$, $r = 20$	96.0	89.8
LRSFN$N_{S_k} = 400$, $r = 20$	91.7	89.3
LRSFN $N_{S_k} = 400$, $r = 30$	91.4	88.9
INCG $N_{S_k} = 400$	100.0	88.74
INCG $N_{S_k} = 1000$	100.0	88.41
INCG $N_{S_k} = 100$	100.0	88.29
LRSFN$N_{S_k} = 800$, $r = 20$	90.0	87.6
LRSFN $N_{S_k} = 1000$, $r = 20$	89.0	87.4
INCG $N_{S_k} = 800$	100.0	86.9
LRSFN$N_{S_k} = 800$, $r = 30$	89.3	86.8
GD	100.0	86.7
LRSFN $N_{S_k} = 1,000$, $r = 30$	86.0	85.9
SFN $N_{S_k} = 1,000$, $r = 30$	83.0	81.7

Table 3.2: Summary of globalization method results for $N_{X_k} = 1,000$

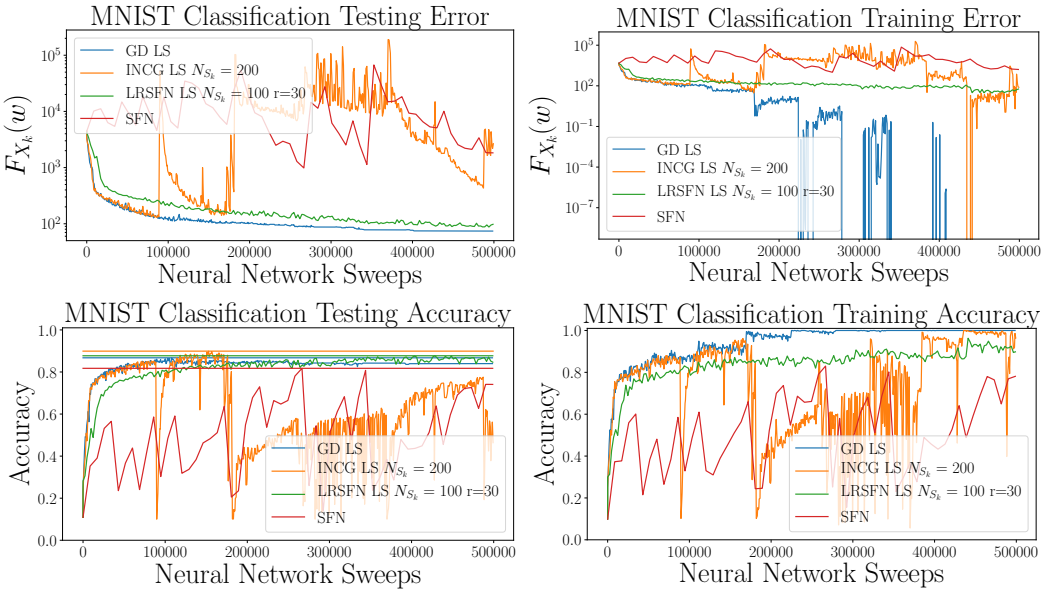


Figure 3.1: MNIST classification results for different optimizers using globalization with $N_{X_k} = 1000$

All methods were given the same initial guess for the weight and the

same partitioning of the training data. Table 3.2 shows that LRSFN reliably performed the best of all of the methods, with INCG as a close second. For this problem LRSFN performed best with the smallest Hessian batch size, but seemed to perform better for the larger rank choice.

Figure 3.1 demonstrates that even with globalization the Krylov methods performed more erratically than LRSFN and GD. I believe this is due to the fact that Krylov methods may incorporate too much noisy information from the Hessian into solves. When a specific instance of a stochastic Hessian has a lot of noise in the Rayleigh quotients noisy components of the gradient may be rescaled poorly and the Newton step may leave a basin of attraction. I believe this explains the spikes for the SFN and INCG method.

For LRSFN it too has some jumps, I believe these are due to the choice of backtracking iterations, not the Hessian noise. When I use 12 backtracking iterations LRSFN takes smaller steps and the spikes are removed. This is seen below in Figure 3.2.

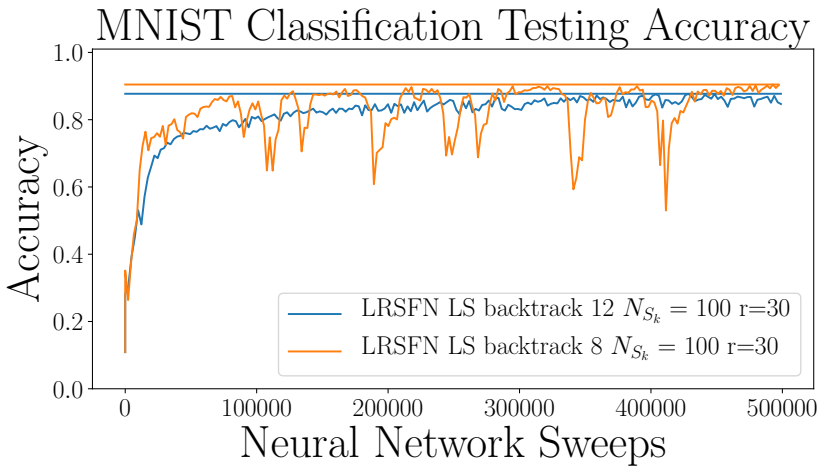


Figure 3.2: LRSFN classification runs for different choice of backtracking parameter $N_{X_k} = 1000$

There is some advantage to these spikes in that it allows the optimizer to explore more of parameter space, and not get stuck in sub-optimal basins. This is shown in Figure 3.2, since more back tracking iterations and less spikes

resulted in worse generalization. How to use this advantage is key however. The LRSFN method outperformed the INCG and SFN method because these spikes were less severe and the successive iterates of LRSFN didn't have to do too much work to return to bottoms of basins. Employing globalization but also seeking to jump between basins may seem counter-intuitive since the goal of globalization is to ensure local convergence. But the stochasticity inherent in the problem makes local convergence difficult; more aggressive searching of parameter space ("basin-hopping") is a practical way to explore many parameter configurations efficiently.

Comparison of methods using fixed steps

In practice methods using fixed step sizes are commonly used to train classification models. This is related to the fact that "basin-hopping" can be s distinctive advantage for an optimizer. Fixed step lengths are agnostic to basins, whereas line search or trust region methods attempt to stay put in one basin. In this section I compare LRSFN and INCG using fixed step lengths against Adam and SGD. I study the effect of rank and step length on the performance of the methods. For INCG and LRSFN I use small Hessian batch sizes $N_{S_k} = 100$ for brevity in the results, as well as increased computational economy.

In the original Adam paper [70], a suggested step size of $\alpha_k = 0.001$ was suggested. For this reason a large range of step sizes are explored for Adam as well as for SGD. For INCG and LRSFN I suggest using larger step sizes, however LRSFN can become unstable for too large a step size. For the second order methods I consider $\alpha_k = 0.01, 0.005, 0.001$, and suggest step sizes in this range for classification problems in general. For LRSFN $\alpha_k = 0.005$ performed the best; I consider many different choices of rank r for this step size. Results are summarized below.

	Train accuracy	Test accuracy
LRSFN $\alpha_k = 0.005$, $r = 20$	100.0	90.9
LRSFN $\alpha_k = 0.005$, $r = 10$	100.0	90.8
LRSFN $\alpha_k = 0.005$, $r = 25$	100.0	90.7
LRSFN $\alpha_k = 0.005$, $r = 40$	99.0	90.5
LRSFN $\alpha_k = 0.005$, $r = 15$	100.0	90.4
SGD $\alpha_k = 0.05$	100.0	90.3
LRSFN $\alpha_k = 0.005$, $r = 30$	100.0	90.2
LRSFN $\alpha_k = 0.005$, $r = 5$	100.0	90.1
Adam $\alpha_k = 0.1$	100.0	89.7
SGD $\alpha_k = 0.1$	99.4	89.6
LRSFN $\alpha_k = 0.005$, $r = 50$	100.0	89.2
Adam $\alpha_k = 0.05$	100.0	89.0
SGD $\alpha_k = 0.01$	100.0	87.9
INCG $\alpha_k = 0.01$	100.0	87.6
LRSFN $\alpha_k = 0.001$, $r = 30$	100.0	87.5
INCG $\alpha_k = 0.005$	100.0	87.2
INCG $\alpha_k = 0.001$	100.0	87.2
Adam $\alpha_k = 0.01$	100.0	85.5
SGD $\alpha_k = 0.005$	100.0	83.8
Adam $\alpha_k = 0.005$	100.0	76.9
SGD $\alpha_k = 0.001$	100.0	75.3
SGD $\alpha_k = 0.0005$	100.0	69.4
SGD $\alpha_k = 0.0001$	98.5	46.1
Adam $\alpha_k = 0.001$	100.0	40.9

Table 3.3: Summary of fixed step results for $N_{X_k} = 1,000$

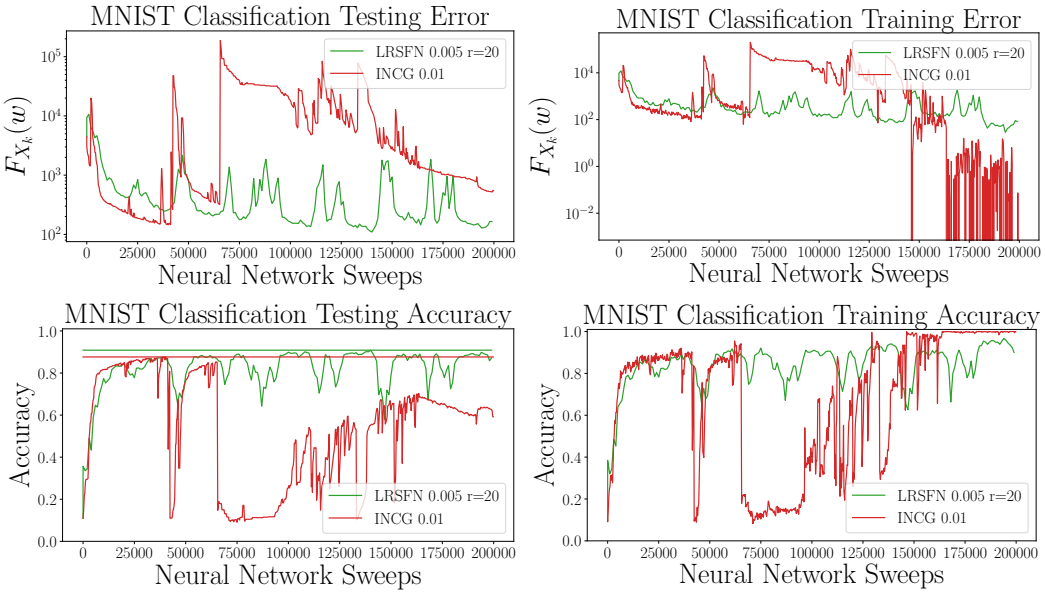


Figure 3.3: MNIST classification best runs for INCG and LRSFN using fixed step lengths with $N_{X_k} = 1000$

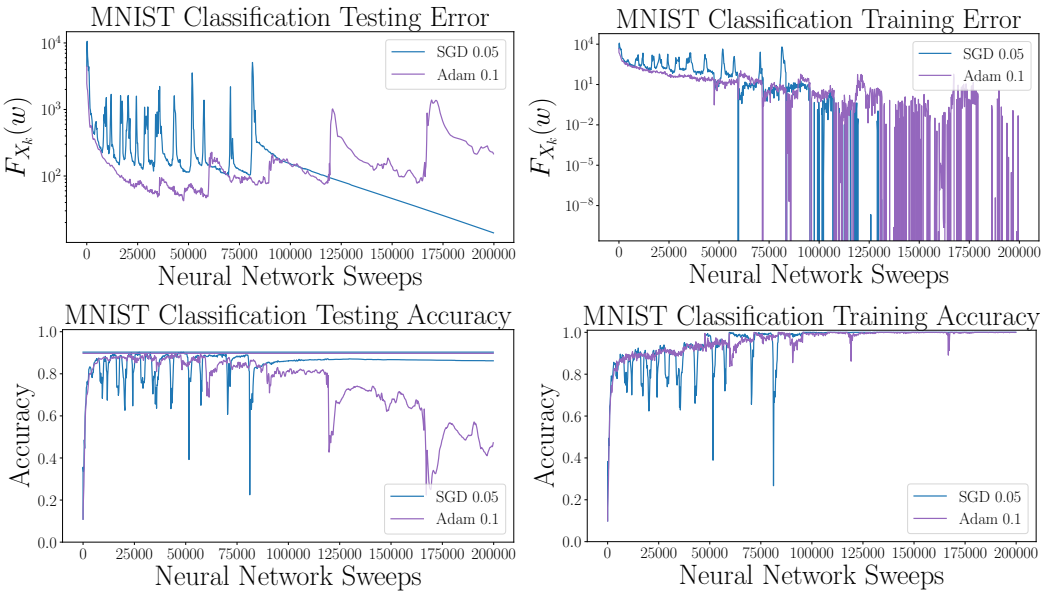


Figure 3.4: MNIST classification best runs for Adam and SGD using fixed step lengths with $N_{X_k} = 1000$

Table 3.3 shows that LRSFN again performed the best compared to all other methods. The choice of $\alpha_k = 0.005$ for LRSFN performed the best

of any method, and for this reason I explore how different ranks performed at this point. All but one of these runs was able to achieve greater than 90% testing accuracy, whereas only one first order method (SGD $\alpha_k = 0.05$) was able to achieve that generalization benchmark. Further the amount of hyper-parameter tuning that may be required to get decent performance from a first order method is evident in Table 3.3. The standard step length choice for Adam achieved merely 40.9% testing accuracy. It is clear that with hyper-parameter tuning one can beat globalization methods, but it may be costly to do so. And it is important to note that several of the LRSFN runs in Table 3.2 were able to clear the 90% testing error threshold. For this reason I advocate for the use of globalized second order methods since they perform reliably well with little to no hyper-parameter tuning required. If one wants to explore fixed step methods, it is clear that second order methods (especially LRSFN) can perform well in this realm.

A common critique of second order methods in the machine learning community is that they are not competitive with first order methods in terms of compute time. The following plot shows that this is not the case for LRSFN, and similar results can be shown for INCG.

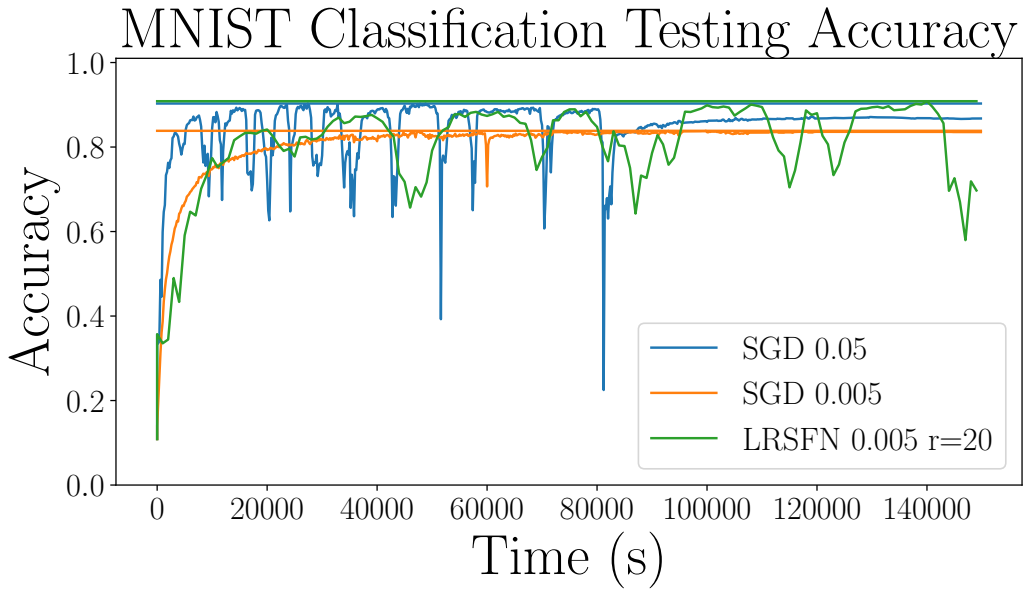


Figure 3.5: MNIST classification timing runs for SGD and LRSFN with $N_{X_k} = 1,000$

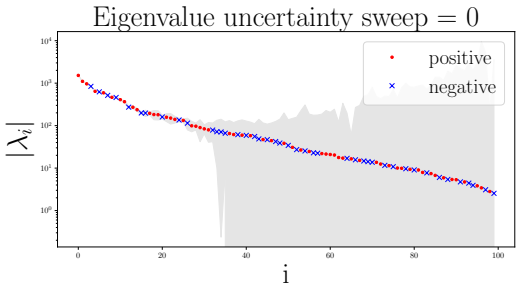
Similar results can be shown for the Newton Krylov methods, which often require very few Hessian vector products at a given iteration in order to satisfy Eisenstat-Walker conditions for convergence.

Now I investigate the uncertainty in the stochastic Rayleigh quotients for the MNIST classification problem.

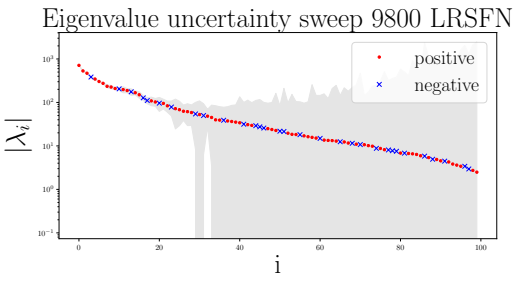
For a single run of INCG and LRSFN with $N_{X_k} = 1,000, N_{S_k} = 100$, as well as SFN with $N_{X_k} = N_{S_k} = 1,000$ I plot the spectrum of the neural network Hessian taken against all of the training data, as well as subsampled Rayleigh quotients that show the eigenvalue noise for indefinite Hessians. This can be seen below in Figure 3.6. The error bars in the plot are the sample average standard deviations of

$$\frac{u_i^T \nabla^2 F_{S_k} u_i}{u_i^T u_i}, \tag{3.6.2}$$

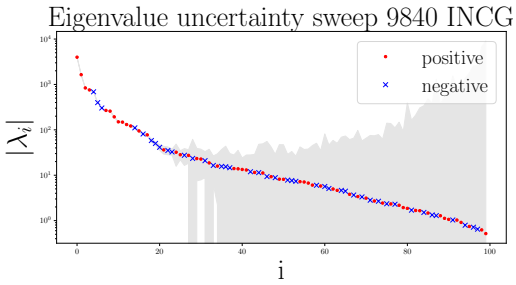
taken for 100 subsets $S_k \subset X$.



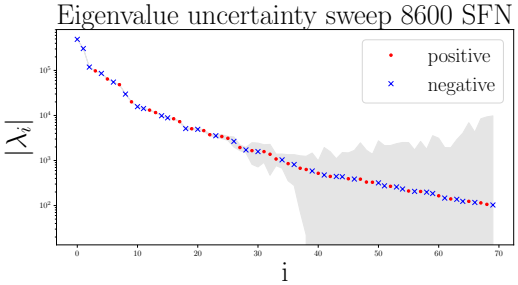
(a) Eigenvalue variance at iteration 0



(b) Eigenvalue variance at iteration 20



(c) Eigenvalue variance at iteration 30



(d) Eigenvalue variance at iteration 40

Figure 3.6: Hessian spectra and uncertainty at different iterations for LRSFN INCG and SFN

There is a high degree of uncertainty at the shared initial guess. At later iterations of INCG and LRSFN the dominant modes tend to be dominated by a few more positive eignevalues. The magnitude of the largest eigenvalues decrease for LRSFN but increase for INCG and SFN. SFN does not perform very well for this problem, and this can be evidenced by the highly indefinite dominant eigenvalues. For this problem which uses a cross-entropy loss function I do not observe regions of positive semi-definiteness until the optimizers begins to overfit. Observations of the cross-entropy Hessian suggest a high degree of indefiniteness and associated eigenvalue noise.

For a second MNIST classification example, I test how fixed step LRSFN performs against first order methods on shallow dense neural networks. In this set of examples, the input data are normalzed to be less than one. This rescaling of the problem makes the scaling of the optimization geometry slightly easier. The dense neural network example that I investigate involves one hidden layer with dimension 50. The configuration space dimension for

this problem is $d_W = 39,760$. There is a much larger configuration space in this example, but not necessarily better representation capabilities. Since the fixed step methods performed the best overall in the earlier example I focus only on fixed step implementations here. I compare LRSFN with Adam and SGD, all using small batch sizes $N_{X_k} = 100$. For LRSFN I use a Hessian batch size of $N_{S_k} = 10$, note that this introduces a high degree of sampling error in the Hessian approximation, but it is justified in some sense since we only seek to resolve modes with very low variance. I study various step sizes for each method, and study different fixed ranks for LRSFN. The results are summarized below.

	Accuracy train	Accuracy test
LRSFN $\alpha_k = 0.01$ $r = 10$	100.0	93.22
LRSFN $\alpha_k = 0.005$ $r = 5$	100.0	92.90
LRSFN $\alpha_k = 0.005$ $r = 20$	100.0	92.55
LRSFN $\alpha_k = 0.005$ $r = 10$	100.0	92.47
LRSFN $\alpha_k = 0.01$ $r = 20$	100.0	92.47
Adam $\alpha_k = 0.01$	100.0	92.29
Adam $\alpha_k = 0.005$	100.0	90.13
Adam $\alpha_k = 0.1$	100.0	88.38
LRSFN $\alpha_k = 0.005$ $r = 25$	100.0	85.04
LRSFN $\alpha_k = 0.01$ $r = 15$	100.0	85.03
LRSFN $\alpha_k = 0.005$ $r = 15$	100.0	84.92
LRSFN $\alpha_k = 0.01$ $r = 25$	100.0	84.63
LRSFN $\alpha_k = 0.01$ $r = 5$	100.0	84.57
Adam $\alpha_k = 0.001$	100.0	69.92
SGD $\alpha_k = 0.1$	100.0	53.71
SGD $\alpha_k = 0.01$	100.0	24.82
Adam $\alpha_k = 0.0001$	100.0	24.62

Table 3.4: Summary of of fixed step results for dense MNIST classifier

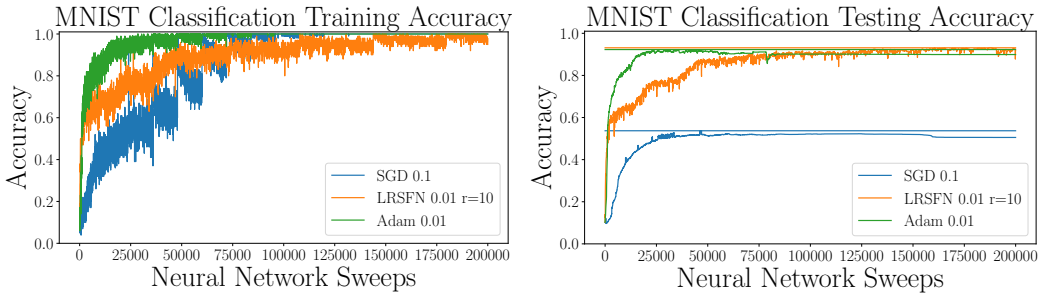


Figure 3.7: MNIST classification best runs for dense classifier

In the last set of numerical examples where convolutional resnet layers were used and the input data were not rescaled, SGD performed best of the SA first order methods. In this case Adam performed better than SGD, but LRSFN outperformed Adam for various choices of the step length and rank. This example demonstrates the robustness of LRSFN to problem scaling in these two modest MNIST classification examples.

I am actively working to scale up to larger problems that are more typical of modern neural networks. So far there are issues with extending these results to very deep purely dense networks. INCG did not perform well on this problem relative to LRSFN, and Newton methods using line search often can exactly predict all of the training data but suffer in generalization error as SGD did in the Table above.

These preliminary results suggest that Newton methods can perform well in classification problems with different scalings, one of the selling points of second order methods in general. In the problems investigated the Newton methods performed the best with LRSFN being the best method in terms of generalizability.

3.6.2 CIFAR10 Convolutional autoencoder

In the second set of numerical experiments, I train convolutional autoencoders on the CIFAR10 data set. For the convolutional autoencoder training problem a least squares loss function is used to measure the error in reconstructing input images with a four layer autoencoder network.

For the first set of results I use $N_{X_k} = 10,000$ for all methods except SGD and Adam which use $N_{X_k} = 1,000$. I use $N_{S_k} = 1,000$ for INCG and LRSFN. For Adam and SGD I test a range of step sizes as in previous results. For LRSFN and SFN I take $r = 20$. The size of the weight space is $d_W = 1,543$. The results can be seen below in Table 3.5. Accuracy here is defined below.

$$\text{Accuracy}_{X_k} = 1 - \frac{1}{N_{X_k}} \sum_{i=1}^{N_{X_k}} \frac{\|y_i - f(x_i, w)\|^2}{\|y_i\|^2} \tag{3.6.3a}$$

	Train accuracy	Test accuracy
LRSFN LS $N_{S_k} = 1000$	63.27	61.20
INCG LS $N_{S_k} = 1000$	65.67	57.08
Adam $\alpha_k = 0.1$	58.67	56.86
SGD $\alpha_k = 0.1$	55.03	51.91
GD LS	63.65	51.51
SFN LS $N_{S_k} = 10,000$	51.31	50.96
SGD $\alpha_k = 0.01$	55.06	50.70
SGD $\alpha_k = 0.001$	36.37	34.20
Adam $\alpha_k = 0.01$	23.07	21.71
SGD $\alpha_k = 0.0001$	5.77	5.43
Adam $\alpha_k = 0.001$	3.80	3.58
Adam $\alpha_k = 0.001$	0.84	0.78

Table 3.5: Summary of results for $N_{X_k} = 10,000$

The following plots summarize how INCG and LRSFN compare against the Adam runs, the GD runs, and SFN.

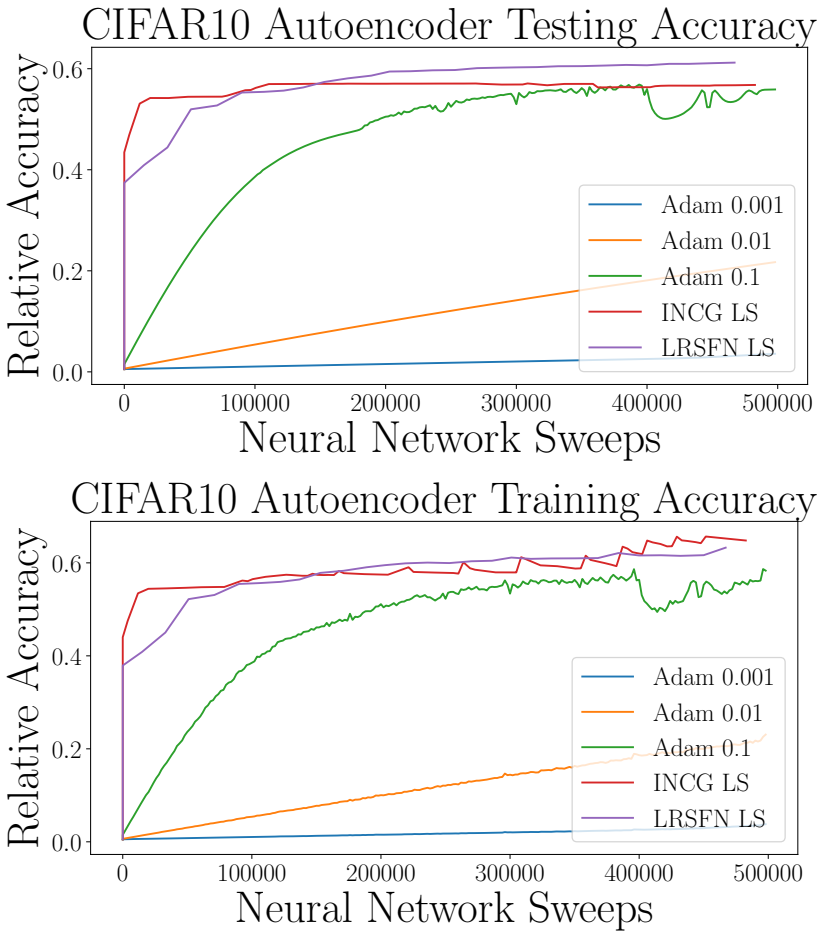


Figure 3.8: CIFAR10 autoencoder results for Adams compared against INCG and LRSFN

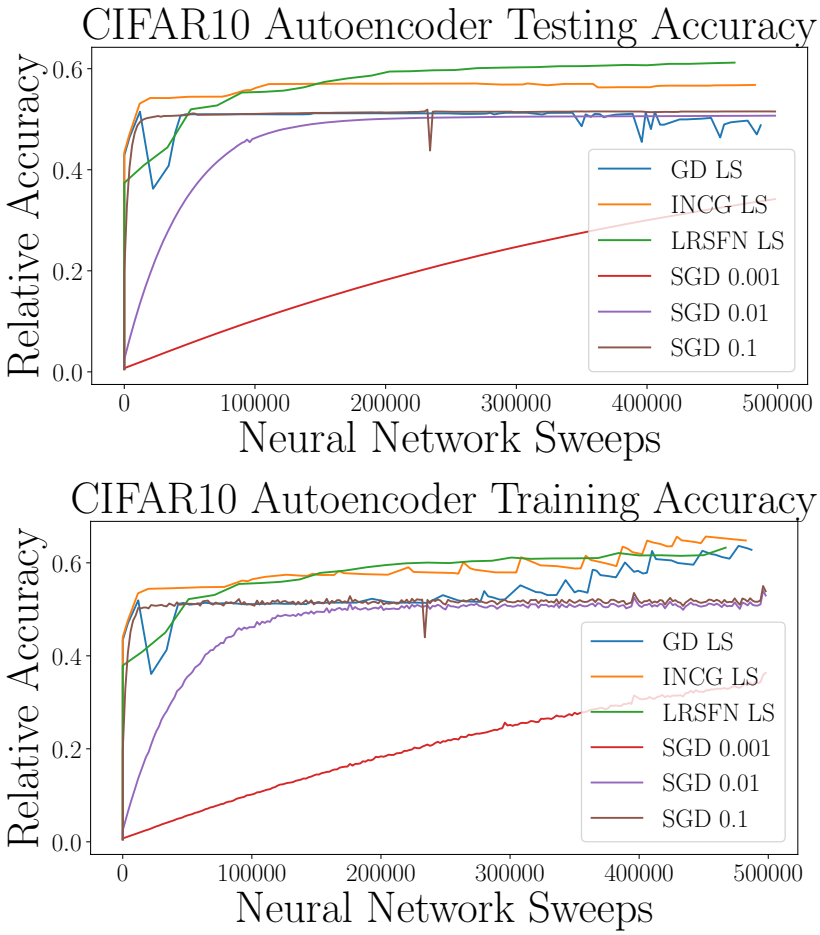


Figure 3.9: CIFAR10 autoencoder results for GDS compared against INCG and LRSFN

As with the MNIST run I observe better performance for smaller batch sizes. In the following I take $N_{X_k} = 1,000$ and study the performance of the methods. The results can be seen below in Table 3.6.

	Train accuracy	Test accuracy
LRSFN LS $N_{S_k} = 100$	78.27	63.50
LRSFN LS $N_{S_k} = 1000$	60.60	60.30
INCG LS $N_{S_k} = 1000$	73.11	58.96
INCG LS $N_{S_k} = 100$	64.98	58.46
GD LS	64.52	58.05
Adam $\alpha_k = 0.1$	58.99	57.06
SGD $\alpha_k = 0.1$	55.05	54.56
SGD $\alpha_k = 0.01$	54.07	53.28
SFN $N_{S_k} = 1000$	53.69	52.73
SGD $\alpha_k = 0.001$	36.37	34.20
Adam $\alpha_k = 0.01$	23.07	21.72
SGD $\alpha_k = 0.0001$	5.77	5.43
Adam $\alpha_k = 0.001$	4.11	3.88
Adam $\alpha_k = 0.001$	0.84	0.78

Table 3.6: Summary of results for $N_{X_k} = 1,000$

Again I compare the performance of the methods against the Adam runs, the GD runs and SFN in the following plots.

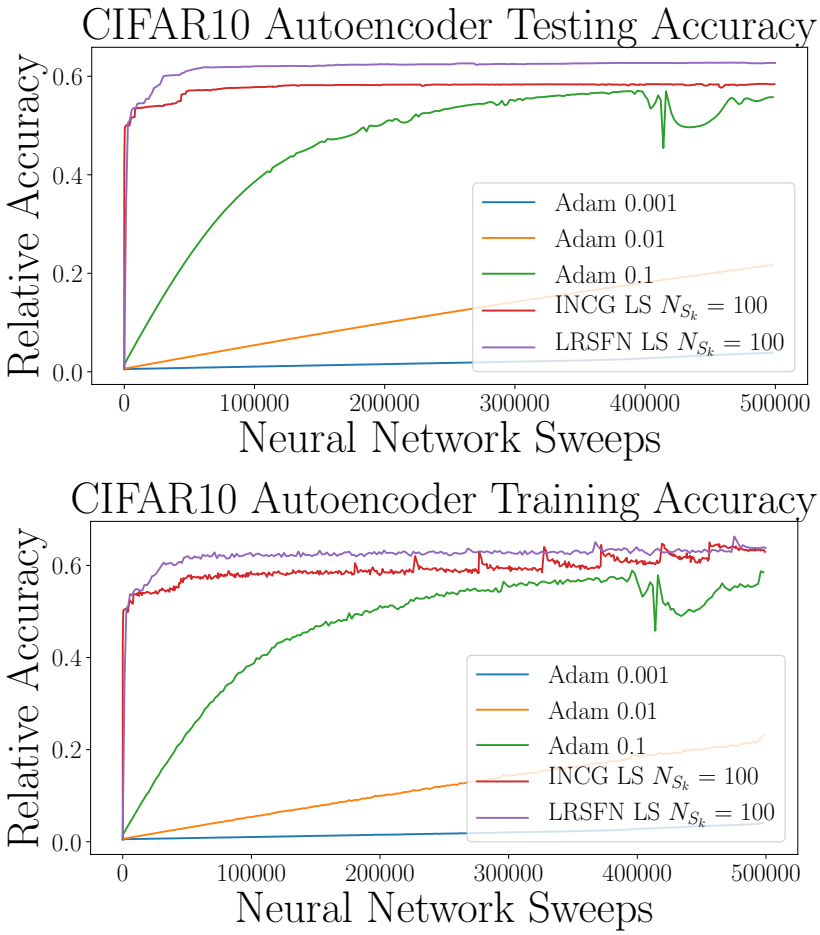


Figure 3.10: CIFAR10 autoencoder results for Adams compared against INCG and LRSFN

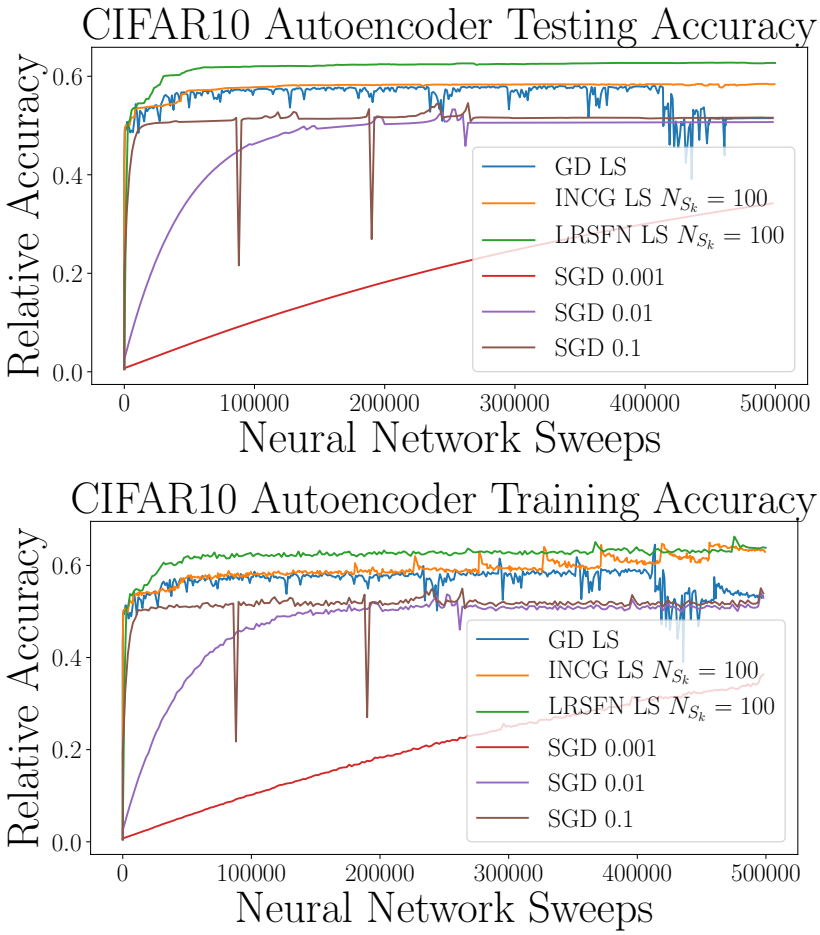


Figure 3.11: CIFAR10 autoencoder results for GDS compared against INCG and LRSFN

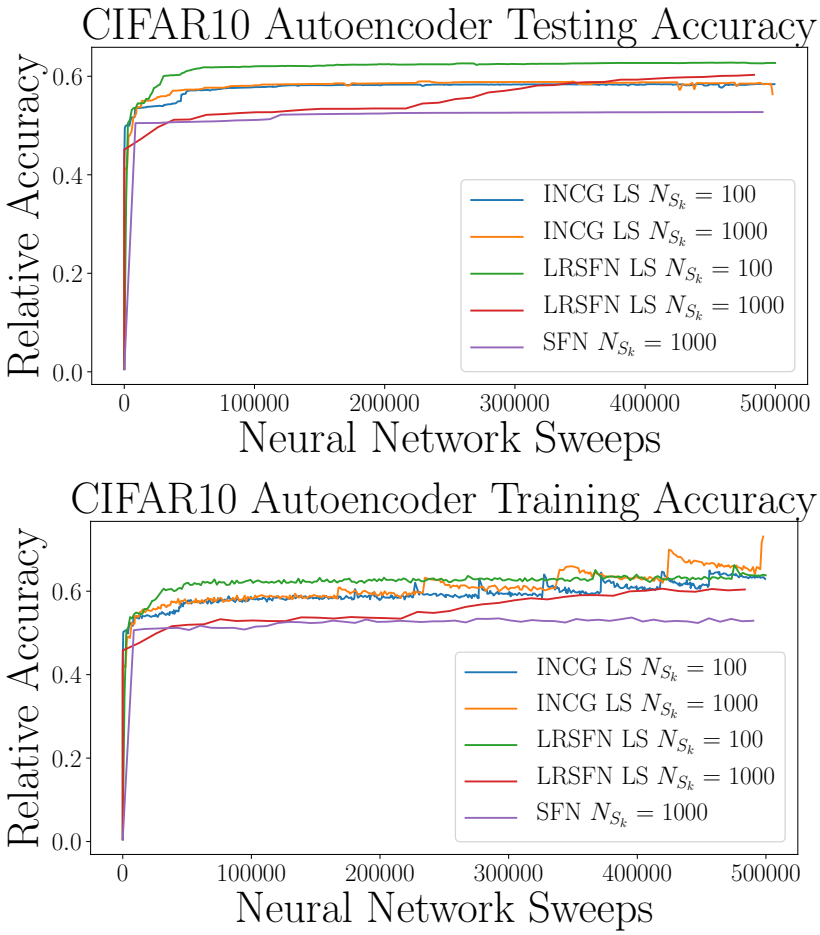


Figure 3.12: CIFAR10 autoencoder results comparison for second order methods

For the autoencoder problem LRSFN and INCG performed better across the board. These methods show significant promise on regression problems. In what follows I investigate the eigenvalue uncertainty for the three second order methods, which sheds light on the performance of the methods. For all methods the initial guess is located in a highly indefinite region of parameter space, with associated noisy eigenvalues. INCG and LRSFN are both able to escape indefinite regions and find regions of positive semi-definiteness. As was predicted by the analysis in Section 3.2 as the spectrum becomes more positive definite, the noisiness associated with indefiniteness seems to subside. When the iterates find a region in parameter space that is positive

semi-definite the noisiness for the eigenvalues goes away. This could also be related to overfitting however.

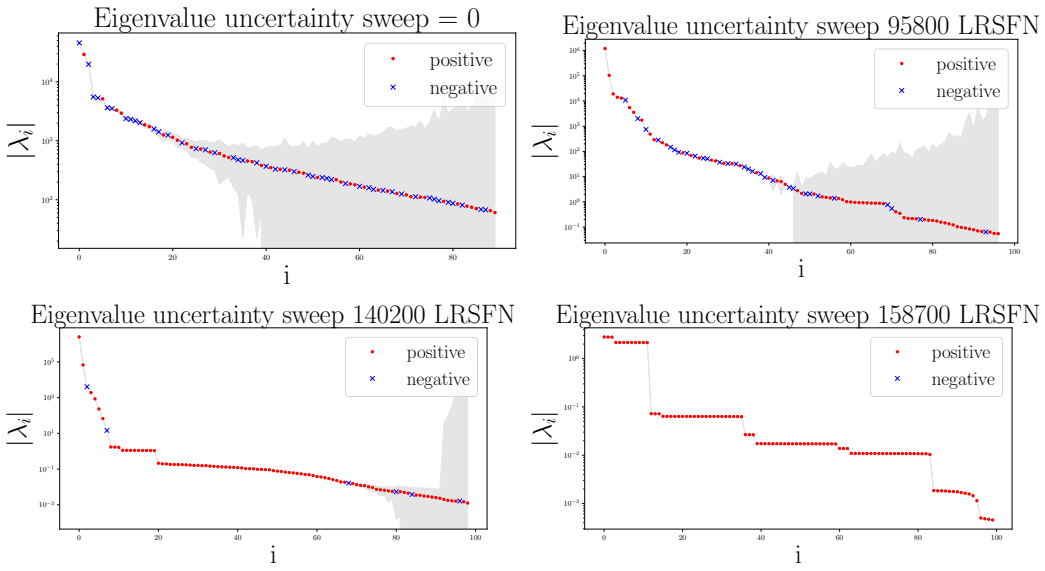


Figure 3.13: CIFAR10 eigenvalue uncertainty for LRSFN

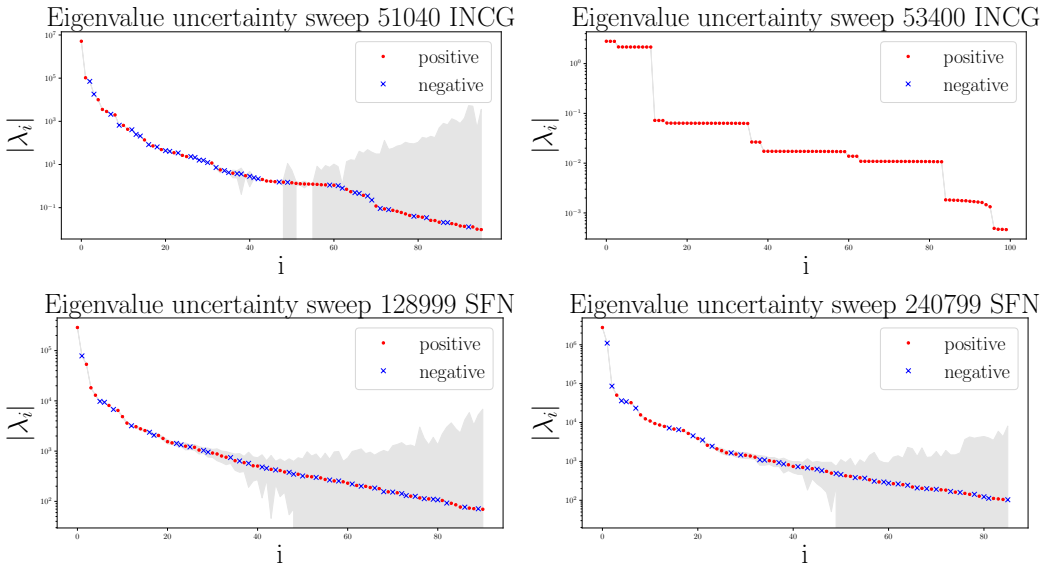


Figure 3.14: CIFAR10 eigenvalue uncertainty for INCG and SFN

3.6.3 Parametric Map Regression

In this section I study how INCG, LRSFN perform on the empirical risk minimization problem for the nonlinear surrogate construction in Chapter 2. The empirical risk minimization problem is a least squares regression problem. I focus specifically on the projected low rank resnet pLRRN architecture that is the focus of the numerical results for that section. I first start with two numerical tests for the convection diffusion problem described in Section 2.4.3. For the first two problems 800 training data are used during training, and 200 data are set aside to evaluate generalization error. I train a pLRRN with projection rank $r = 40$, four nonlinear layers, each with inner layer rank $r = 12$. The nonlinear activation functions used are softmax for the first layers, followed by softplus and identity for the last layers. A more thorough discussion of this architecture can be found in Section 2.2.

The first two datasets considered are on a mesh with $n_x = n_y = 96$. The input data have dimension $9,409$, the output data have dimension 100. The neural network used here has configuration space dimension $d_W = 5,140$. For INCG and LRSFN I used line search (LS). I compare against Adam and SGD which require hyperparameter tuning to find an appropriate step size α_k. I consider two different parametrizations of the Matern covariance prior used for the parametric map $\gamma, \delta = 0.5, 0.5$ and $\gamma, \delta = 2.0, 0.5$ (see Section 2.4.1 for more information). The results are summarized below, note that since accuracy is defined as

$$100 \left(1 - \frac{\sqrt{\sum_{i=1}^{N_{\text{test}}} \|y_i - f(x_i, w)\|^2}}{\sqrt{\sum_{i=1}^{N_{\text{test}}} \|y_i\|^2}} \right), \tag{3.6.4}$$

it can be negative when an approximation is very bad.

	Accuracy train	Accuracy test
LRSFN LS	39.41	37.11
INCG LS	70.99	32.01
GD LS	70.70	18.17
Adam $\alpha_k = 0.1$	64.82	17.65
Adam $\alpha_k = 0.01$	-365.66	-137.22
SGD $\alpha_k = 0.01$	-408.44	-296.56
SGD $\alpha_k = 0.001$	-420.87	-415.98
Adam $\alpha_k = 0.001$	-581.41	-450.12
SGD $\alpha_k = 0.1$	-798.61	-1061.90

Table 3.7: Summary of training results for pLRRN $r = 40$, $\gamma, \delta = 0.5, 0.5$

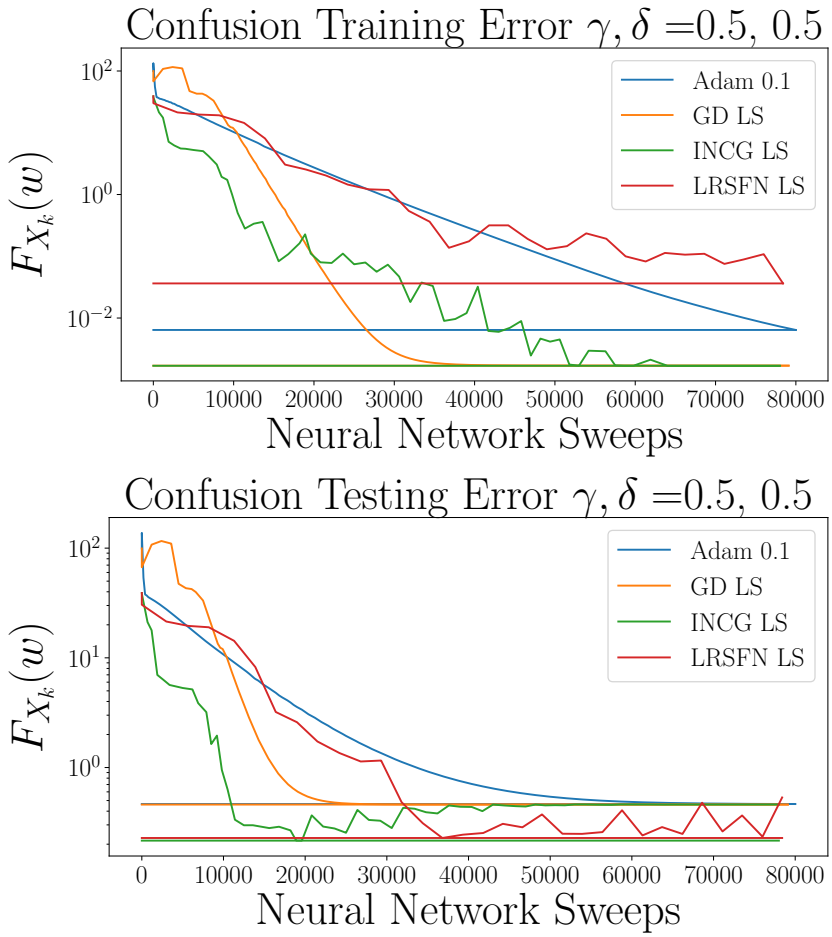


Figure 3.15: Training and Testing Errors for $\gamma, \delta = 0.5, 0.5$

For the first dataset $\gamma = \delta = 0.5$ LRSFN and INCG perform the best of

all the methods by a large margin, with LRSFN outperforming INCG. GD with LS outperformed all of the other first order methods. Adam and SGD were sensitive to the choice of step length α_k, and were unstable for α_k large. This suggests that the optimization problem may have a restrictive Lipschitz constant. All of the methods except for LRSFN overfitted significantly; they found weights that performed significantly better on the training data than on the testing data. Similar trends can be seen for the following example. Plots of eigenvalue uncertainty shown in Figure 3.16 demonstrate a link between eigenvalue uncertainty and infefiniteness.

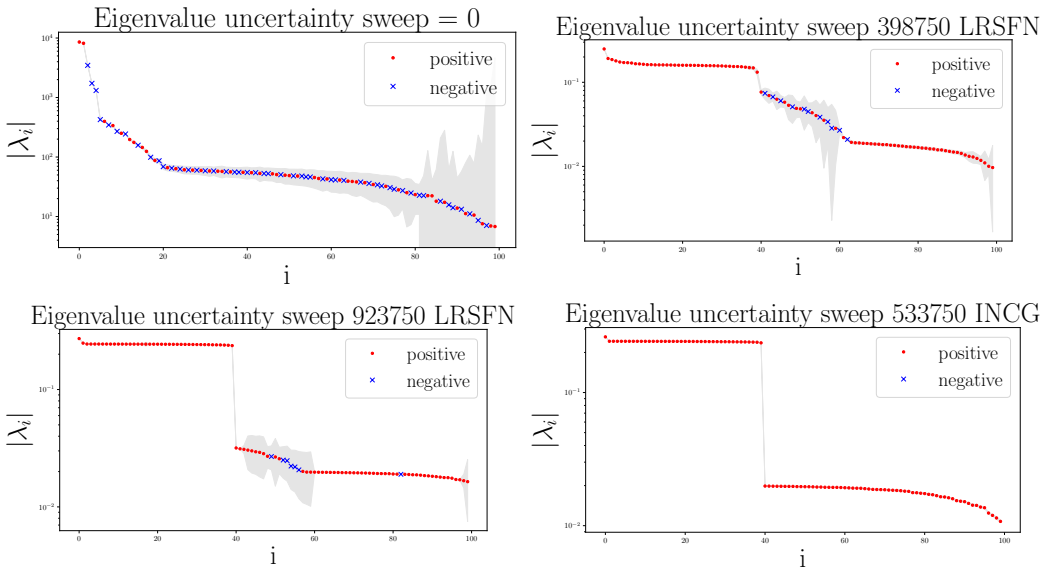


Figure 3.16: Confusion $\gamma = \delta = 0.5$ eigenvalue uncertainty for INCG and LRSFN

	Accuracy train	Accuracy test
LRSFN LS	41.66	33.28
INCG LS	75.81	11.08
GD LS	75.80	10.80
Adam $\alpha_k = 0.1$	52.94	10.31
Adam $\alpha_k = 0.01$	-620.18	-401.20
SGD $\alpha_k = 0.01$	-641.77	-525.15
Adam $\alpha_k = 0.001$	-806.83	-684.09
SGD $\alpha_k = 0.001$	-675.62	-700.65
SGD $\alpha_k = 0.1$	-934.85	-1211.23

Table 3.8: Summary of training results for pLRRN $r = 40$, $\gamma, \delta = 2.0, 0.5$

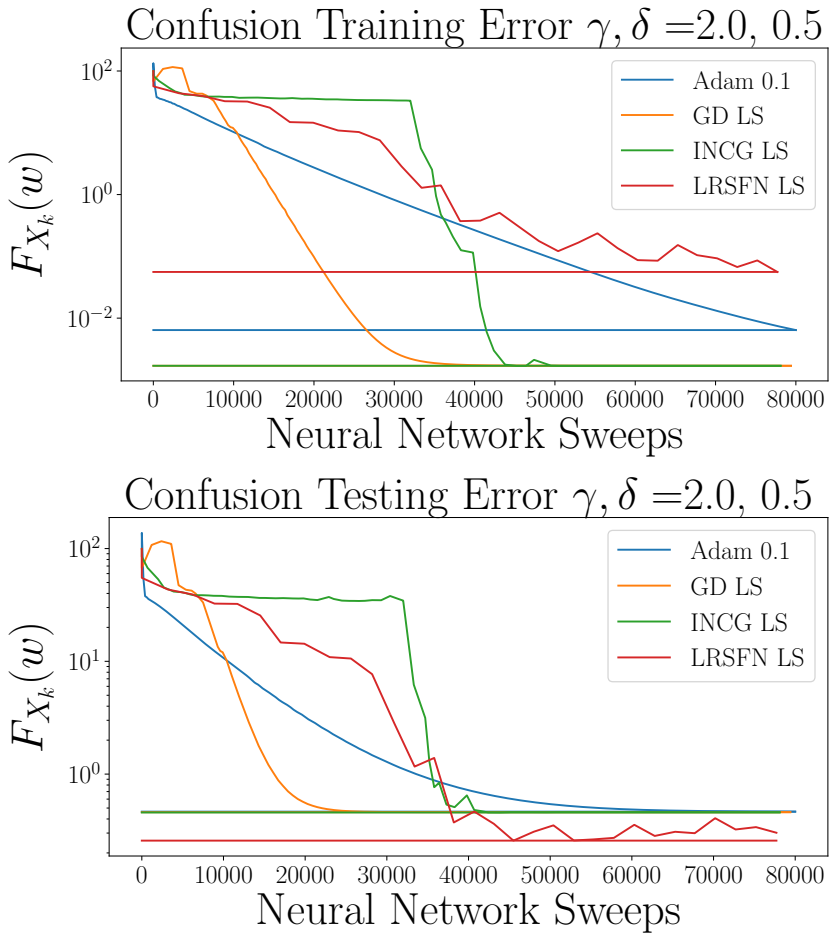


Figure 3.17: Training and Testing Errors for $\gamma, \delta = 2.0, 0.5$

In this case LRSFN significantly outperformed all other methods. All of

the other methods found weights that generalized poorly. Again the first order methods were very sensitive to hyperparameter tuning. Plots of eigenvalue uncertainty shown in Figure 3.18 demonstrate again a link between eigenvalue uncertainty and infefiniteness.

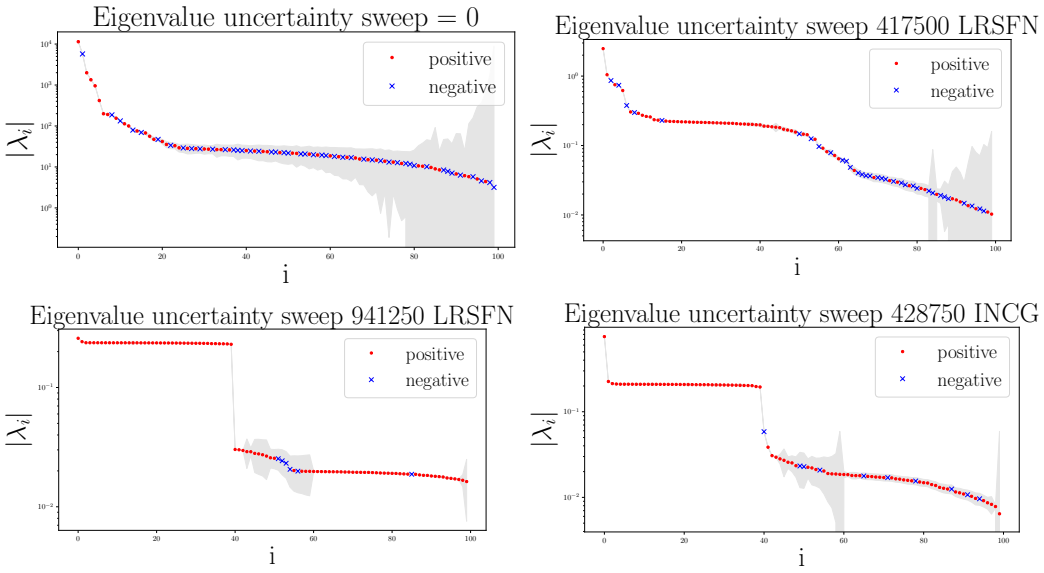


Figure 3.18: Confusion $\gamma = 2.0, \delta = 0.5$ eigenvalue uncertainty for INCG and LRSFN

3.6.4 Software

Implementations of LRSFN and INCG can be found at `https://github.com/tomoleary/hessianlearn`, a Python library for second order stochastic optimization in TensorFlow [1].

3.7 Conclusion

In this work I analyze the use of matrix-free Newton methods in stochastic nonconvex optimization problems.

Newton methods are typically considered uncompetitive in modern machine learning problems due to the expense of computing the Hessian inverse. Approximate Newton algorithms that only require the Hessian action can achieve good performance for computational work that is comparable to

that of subsampled first order methods. I specifically am interested in how subsampled Newton methods can be efficiently designed to achieve good convergence and generalizability.

I start by analyzing the effect of subsampling on the variance of eigenvalues in the stochastic Hessians used in subsampled Newton methods. For a statistical model, eigenvalue approximation error by subsampled Hessians had unbounded variance in regions where the true stochastic Hessian has eigenvalues that change sign. This is the case when iterates for the empirical risk minimization are leaving a locally indefinite region and entering a local positive semi-definite region. This analysis suggests that in indefinite regions the lower order modes of the Hessian could exhibit high variance, but larger magnitude eigenvalues (directions which when locally perturbed do not change sign) could exhibit lower variance.

Smaller eigenvalues of subsampled Hessians in numerical experiments exhibited very large amounts of variance, agreeing with the conjecture based on the analysis for the statistical model. This result suggests that only the dominant modes of the Hessian are certain, and when the Hessian is indefinite, smaller eigenvalues can exhibit large variance, i.e. be dominated by statistical noise.

The first Newton methods I investigate are Newton Krylov methods. Krylov methods iteratively build a solution in a way that is equivalent to minimizing an error represented as a polynomial function of the eigenvalues of the corresponding matrix. These minimal polynomials incorporate information about the entire spectrum of the matrix, and accordingly Krylov methods are optimal in the sense that they pay attention to the entire matrix spectrum. In the vicinity of a local minimum, Krylov methods can achieve fast convergence. When the Hessian spectrum has fast decay or clusters, then the Newton Krylov solve can converge in only a few iterations, since only a low order polynomial can have zeros near the dominant modes of the Hessian spectrum. If the problem is indefinite, early termination techniques

can be used to avoid taking steps in ascent directions, but nothing can be done to avoid using potentially noisy lower order modes of the subsampled Hessian spectrum that may be statistical noise.

This motivates the low rank saddle free Newton algorithm (LRSFN). In the vicinity of saddle points, full Newton can rescale components pointing away from the saddle back towards the saddle point. This can be alleviated by using a spectral decomposition of the Hessian matrix and taking the absolute values of all of the eigenvalues.

Eigenvalue problems for a stochastic Hessian matrix formally require $O(d_W^2)$ formation cost, and $O(d_W^3)$ computation for the factorization. Factorizations can be approximated for $O(d_W r)$ formation cost and $O(d_W r^2)$ computation for the factorization. In the SFN method [41] a Lanczos procedure is used to approximate the dominant modes of the Hessian.

In this work I propose using a randomized low rank factorization of the Hessian that can be efficiently inverted using a Sherman Morrison Woodbury formula when regularization or Levenberg-Marquardt damping are employed. The randomized low rank eigenvalue decomposition can efficiently approximate the top r eigenvalues of a Hessian matrix in an efficient scalable way.

The computational kernel used in the Hessian approximation can be easily parallelized in a way inherently sequential Krylov methods cannot. In addition to being inherently sequential, low rank is optimal in the sense that it attempts to only pay attention to the top modes of the Hessian matrix, which happen to be the persistent modes of the stochastic Hessian. So even in the case that the Hessian is not low rank, the low rank approximation can still be useful to speed up convergence in some directions. If the Hessian matrix has fast spectral decay in the viscinity of a local minimum, then LRSFN can achieve fast convergence for a small amount of computational work.

I compare the LRSFN and Inexact Newton CG (INCG) methods against the SFN algorithm and first order methods Adam, gradient descent, stochastic gradient descent etc. on three problems. For MNIST classification LRSFN

performed better than the other methods for a problem involving a convolutional residual neural network (resnet) where stochastic gradient descent performed well, and a shallow dense neural network where Adam performed well. Experimental evidence suggested that only the top modes of the Hessian eigenvalues could be well approximated when the Hessian was indefinite. LRSFN was parametrized to only attempt to resolve these modes and was able to find more generalizable solutions than all of the other methods for these problems.

INCG and LRSFN performed better than all other methods on the two regression problems studied in this thesis. The first was a CIFAR10 autoencoder problem, the second was a problem arising from the construction of a surrogate for a nonlinear parametric mapping (see Chapter 2).

The LRSFN algorithm demonstrates that matrix-free Newton methods can use spectral properties of stochastic Hessians (decay and variance for indefinite regions) to build efficient Newton methods that are tuned to the noise of the problem, and lead to better generalization. INCG can perform almost as well in some problems when it is modified to handle indefiniteness.

Low rank approximations of Hessian information has been useful in other contexts such as high dimensional Bayesian inversion [28, 32, 35]. Efficiently parametrized Newton methods that take into account the noise of the problem can outperform first order methods; these methods should be used more in practice.

In future work I am looking to parallelize the LRSFN algorithm and extend these results to larger dimensional neural networks and bigger data sets.

3.8 Convergence Bounds

3.8.1 Convergence of Subsampled Inexact Newton Methods with Eisenstat-Walker

Theorem 3.2 (Local convergence for stochastic inexact Newton methods with gradient norm forcing)**.** *Let w^* be a stationary point and suppose that assumptions A1-A4 hold, let*

$$\mu = \min\left\{\|\nabla^2 F(w^*) + \gamma I\|^{-1}, \|[\nabla^2 F(w^*) + \gamma I]^{-1}\|\right\}, \tag{3.3.2}$$

and assume that

(a) $w_k \in B_\delta(w^*)$ *with* $\delta < \frac{2\mu}{L_{N_{S_k}}}$,

(b) $-\epsilon_H I \preceq \nabla^2 F_{S_k}$ *for all* S_k *and for all* $w \in B_\delta(w^*)$,

(c) The Tikhonov regularization parameter is chosen such that $\gamma > \epsilon_H$,

(d) $\|\nabla^2 \overline{F}_{S_k}(w_k)p_k - \nabla \overline{F}_{X_k}(w_k)\| \leq \eta_k \|\nabla \overline{F}_{X_k}(w_k)\|$ *with* $\eta_k \leq \|\nabla \overline{F}_{X_k}(w_k)\|$.

Then for the iterate $w_{k+1} = w_k + \alpha_k p_k$*, one has the following bound:*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2, \tag{3.3.3}$$

where

$$c_0 = \frac{1}{\gamma - \epsilon_H}\left[\frac{\alpha_k v}{\sqrt{N_{X_k}}}\left(1 + \frac{v}{\sqrt{N_{X_k}}}\right)\right] \tag{3.3.4a}$$

$$c_1 = \frac{1}{\gamma - \epsilon_H}\left(L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} + \frac{2\alpha_k v\mu}{\sqrt{N_{X_k}}}\right) \tag{3.3.4b}$$

$$c_2 = \frac{1}{\gamma - \epsilon_H}\left(\frac{M}{2} + \alpha_k \mu^2\right). \tag{3.3.4c}$$

Proof. I begin by expanding the left hand side of equation 3.3.3 and employ-

ing the triangle inequality:

$$\begin{aligned}
&\mathbb{E}_k[\|w_{k+1} - w^*\|] \\
&= \mathbb{E}_k[\|w_k - w^* - \alpha_k p_k\|] \\
&= \mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)^{-1}(\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) \\
&\quad - \alpha_k \nabla^2 \overline{F}_{S_k}(w_k) p_k + \alpha_k \nabla \overline{F}_{X_k}(w_k) - \alpha_k \nabla \overline{F}_{X_k}(w_k))\|] \\
&\leq \frac{1}{\gamma - \epsilon_H} \underbrace{\mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla \overline{F}_{X_k}(w_k)\|]}_{\text{term 1}} \\
&\quad + \frac{\alpha_k}{\gamma - \epsilon_H} \underbrace{\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla^2 \overline{F}_{S_k}(w_k) p_k\|]}_{\text{term 2}}. \qquad (3.8.1)
\end{aligned}$$

Term 1 can be bounded as

$$\begin{aligned}
&\frac{1}{\gamma - \epsilon_H} \mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla \overline{F}_{X_k}(w_k)\|] \\
&\leq \frac{1}{\gamma - \epsilon_H} \left[\frac{M}{2} \|w_k - w^*\|^2 + \left(L_{N_{S_k}} |1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} \right) \|w_k - w^*\| + \frac{\alpha_k v}{\sqrt{N_{X_k}}} \right]. \qquad (3.8.2)
\end{aligned}$$

By Lemma 3.2. For term 2, by assumption

$$\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla^2 \overline{F}_{S_k}(w_k) p_k\|] \leq \mathbb{E}_k[\eta_k \|\nabla \overline{F}_{X_k}(w_k)\|] \leq \mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k)\|^2]. \qquad (3.8.3)$$

By a bound given in Theorem 2.1 in [15] and the Monte Carlo approximation error:

$$\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla \overline{F}(w_k)\|^2] \leq \frac{v^2}{N_{X_k}}. \qquad (3.8.4)$$

Employing the reverse triangle inequality one can obtain

$$\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k)\|^2] \leq \left(\|\nabla \overline{F}(w_k)\| + \frac{v}{\sqrt{N_{X_k}}} \right)^2. \qquad (3.8.5)$$

By Lemma 1.2 in [43],

$$\|\nabla \overline{F}(w_k)\| \leq \mu \|w_k - w^*\|. \qquad (3.8.6)$$

So combining equations (3.8.5) and (3.8.6) one can obtain the bound

$$\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k)\|^2] \leq \mu^2 \|w_k - w^*\|^2 + \frac{2v\mu}{\sqrt{N_{X_k}}} \|w_k - w^*\| + \frac{v^2}{N_{X_k}}. \qquad (3.8.7)$$

□

3.8.2 Convergence of Inexact Newton Krylov Methods

I begin with a few Lemmas.

Lemma 3.1. *Suppose that assumptions A1 and A3 hold and $\alpha_k > 0$. Then the following bound holds:*

$$\|\nabla^2 F(w_k)(w_k - w^*) - \alpha_k \nabla F(w_k)\| \leq \frac{M}{2}\|w_k - w^*\|^2 + L_{N_{S_k}}|1-\alpha_k|\|w_k - w^*\|. \tag{3.8.8}$$

Proof. The triangle inequality allows us to split the left hand side of (3.8.8) as follows:

$$\begin{aligned}
&\|\nabla^2 F(w_k)(w_k - w^*) - \alpha_k \nabla F(w_k)\| \\
&= \|\nabla^2 F(w_k)(w_k - w^*) - \nabla F(w_k) + (1-\alpha_k)\nabla F(w_k)\| \\
&\leq \underbrace{\|\nabla^2 F(w_k)(w_k - w^*) - \nabla F(w_k)\|}_{\text{term 1}} + |1-\alpha_k| \underbrace{\|\nabla F(w_k)\|}_{\text{term 2}}.
\end{aligned} \tag{3.8.9}$$

By a derivation in Lemma 2.2 in [15] that uses the Lipschitz continuity of the Hessian, one can bound term 1 by

$$\|\nabla^2 F(w_k)(w_k - w^*) - \nabla F(w_k)\| \leq \frac{M}{2}\|w_k - w^*\|^2. \tag{3.8.10}$$

Defining $h(t) = \nabla F(w^* + t(w_k - w^*))$, one may bound term 2 as follows:

$$\begin{aligned}
\|\nabla F(w_k)\| &= \big|\|\nabla F(w_k)\| - \|\nabla F(w^*)\|\big| \\
&\leq \|\nabla F(w_k) - \nabla F(w^*)\| \\
&= \|h(1) - h(0)\| \\
&= \|\int_0^1 h'(t)dt\| \\
&\leq \int_0^1 \|h'(t)\|dt \\
&= \int_0^1 \|\nabla^2 F(w^* + t(w_k - w*))(w_k - w^*)\|dt \\
&\leq \int_0^1 L_{N_{S_k}}\|w_k - w^*\|dt \\
&= L_{N_{S_k}}\|w_k - w^*\|.
\end{aligned} \tag{3.8.11}$$

□

Lemma 3.2. *Suppose that assumptions A1-A4 hold, and $\alpha_k > 0$, then*

$$\mathbb{E}_k[\|\nabla^2 F_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla F_{X_k}(w_k)\|] \leq$$
$$\frac{M}{2}\|w_k - w^*\|^2 + \left(L_{N_{S_k}}|1-\alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}}\right)\|w_k - w^*\| + \frac{\alpha_k v}{\sqrt{N_{X_k}}}. \quad (3.8.12)$$

Proof. This result follows immediately from Lemma 3.1 and Lemmas 2.2 and 2.3 in [15]. □

Theorem 3.3 (Local convergence of stochastic inexact Newton CG (INCG), extension of Lemma 3.1 of [15])**.** *Let w^* be a stationary point, suppose assumptions A1-A4 hold, and the iterates $\{w_k\}$ are generated by the stochastic inexact Newton CG method, the direction p_k^r is found in $r \ll d$ steps (for justification see section 3.4.2), and there exists $\epsilon_H > 0$ such that $-\epsilon_H I \preceq \nabla^2 F_{S_k}(w_k)$ and $\gamma > \epsilon_H$. Then,*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2 \quad (3.4.3)$$

where

$$c_0 = \frac{\alpha_k v}{(\gamma - \epsilon_H)\sqrt{N_{X_k}}} \quad (3.4.4a)$$

$$c_1 = \frac{1}{\gamma - \epsilon_H}\left[L_{N_{S_k}}|1-\alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} + 2\alpha_k L_{N_{S_k}}\sqrt{\kappa_{N_{S_k}}}\left(\frac{\sqrt{\kappa_{N_{S_k}}}-1}{\sqrt{\kappa_{N_{S_k}}}+1}\right)^r\right] \quad (3.4.4b)$$

$$c_2 = \frac{M}{2(\gamma - \epsilon_H)}, \quad (3.4.4c)$$

and $\kappa_{N_{S_k}}$ is the condition number of the Tikhonov regularized Hessian.

Proof. Expanding the left hand side of equation (3.4.3) and employing the triangle inequality, the following bound can be established:

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] = \mathbb{E}_k[\|w_k - w^* + \alpha_k p_k^r\|]$$
$$\leq \underbrace{\mathbb{E}_k[\|w_k - w^* - \alpha_k \nabla^2 \overline{F}_{S_k}(w_k)\nabla \overline{F}_{X_k}(w_k)\|]}_{\text{term 1}} + \alpha_k \underbrace{\mathbb{E}_k[\|p_k^r - \nabla^2 \overline{F}_{S_k}^{-1}(w_k)\nabla \overline{F}_{X_k}\|]}_{\text{term 2}}. \quad (3.8.13)$$

The first term can be bounded as

$$\mathbb{E}_k[\|w_k - w^* - \alpha_k \nabla^2 \overline{F}_{S_k}(w_k) \nabla \overline{F}_{X_k}(w_k)\|] \leq$$
$$\frac{1}{\gamma - \epsilon_H} \mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla \overline{F}_{X_k}(w_k)\|] \leq$$
$$\frac{1}{\gamma - \epsilon_H}\left[\frac{M}{2}\|w_k - w^*\|^2 + \left(L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}}\right)\|w_k - w^*\| + \frac{\alpha_k v}{\sqrt{N_{X_k}}}\right], \tag{3.8.14}$$

by Lemma 3.2. Term 2 can be bounded by the worst case convergence of the CG algorithm as in Lemma 3.1 in [15]

$$\mathbb{E}_k[\|p_k^r - \nabla^2 \overline{F}_{S_k}^{-1}(w_k) \nabla \overline{F}_{X_k}\|] \leq 2\alpha_k \frac{L_{N_{S_k}}}{\gamma - \epsilon_H} \sqrt{\kappa_{N_{S_k}}} \left(\frac{\sqrt{\kappa_{N_{S_k}}} - 1}{\sqrt{\kappa_{N_{S_k}}} + 1}\right)^r \|w_k - w^*\|. \tag{3.8.15}$$

□

Theorem 3.6 (Local convergence of stochastic inexact Newton GMRES and MINRES)**.** *Let w^* be a stationary point, suppose that assumptions A1-A5 hold, additionally, for some $\delta > 0$, $-\epsilon_H I \preceq \nabla^2 F_{S_k}$ for all S_k and for all $w \in B_\delta(w^*)$ and $\gamma > \epsilon_H$, and the direction p_k^r is found in $r \ll d$ steps. Then one can obtain the following expected value convergence rate bound for the iterates of the stochastic inexact Newton GMRES/MINRES methods:*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2 \tag{3.8.16}$$

where

$$c_0 = \frac{\alpha_k}{\gamma - \epsilon_H}\left(\frac{v}{\sqrt{N_{X_k}}} + \frac{\epsilon_g}{(\gamma - \epsilon_H)}\mathcal{E}\right) \tag{3.8.17a}$$

$$c_1 = \frac{1}{\gamma - \epsilon_H}\left(L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}} + \frac{\alpha_k L_{N_{X_k}}}{(\gamma - \epsilon)}\mathcal{E}\right) \tag{3.8.17b}$$

$$c_2 = \frac{M}{2(\gamma - \epsilon_H)}. \tag{3.8.17c}$$

When the GMRES solver is used,

$$\mathcal{E} = \frac{L_{N_{S_k}}}{\gamma - \epsilon_H} \frac{C_r(\frac{a}{d})}{|C_r(\frac{c}{d})|} \tag{3.8.18a}$$

$$a = (L_{N_{S_k}} - \gamma + \epsilon_H) + 2\epsilon, \quad c = \frac{1}{2}(L_{N_{S_k}} + \gamma - \epsilon_H), \quad d = \frac{1}{2}(L_{N_{S_k}} - \gamma + \epsilon_H) \tag{3.8.18b}$$

and C_r is the r^{th} order Chebyshev polynomial. For MINRES,

$$\mathcal{E} = \left(1 - \frac{(\gamma - \epsilon_H)^2}{L_{N_{S_k}}^2}\right)^{\frac{r}{2}}. \tag{3.8.19}$$

Proof. Expanding the left hand side of equation (3.8.16) and employing the triangle inequality, the following bound can be established:

$$\begin{aligned}
&\mathbb{E}_k[\|w_{k+1} - w^*\|] \\
&= \mathbb{E}_k[\|w_k - w^* - \alpha_k p_k^r\|] \\
&= \mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)^{-1}(\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) \\
&\quad - \alpha_k \nabla^2 \overline{F}_{S_k}(w_k) p_k^r + \alpha_k \nabla \overline{F}_{X_k}(w_k) - \alpha_k \nabla \overline{F}_{X_k}(w_k))\|] \\
&\leq \frac{1}{\gamma - \epsilon_H} \underbrace{\mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla \overline{F}_{X_k}(w_k)\|]}_{\text{term 1}} \\
&\quad + \frac{\alpha_k}{\gamma - \epsilon_H} \underbrace{\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla^2 \overline{F}_{S_k}(w_k) p_k^r\|]}_{\text{term 2}}
\end{aligned} \tag{3.8.20}$$

Term 1 can be bounded as

$$\begin{aligned}
&\frac{1}{\gamma - \epsilon_H} \mathbb{E}_k[\|\nabla^2 \overline{F}_{S_k}(w_k)(w_k - w^*) - \alpha_k \nabla \overline{F}_{X_k}(w_k)\|] \leq \\
&\frac{1}{\gamma - \epsilon_H} \left[\frac{M}{2}\|w_k - w^*\|^2 + \left(L_{N_{S_k}}|1 - \alpha_k| + \frac{\sigma}{\sqrt{N_{S_k}}}\right)\|w_k - w^*\| + \frac{\alpha_k v}{\sqrt{N_{X_k}}}\right].
\end{aligned} \tag{3.8.21}$$

By Lemma 3.2. For term 2 one can use a generic Krylov residual error bound

$$\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla^2 \overline{F}_{S_k}(w_k) p_k^r\|] \leq \mathcal{E}\mathbb{E}_k[\|\nabla \overline{F}_{X_k}\|]. \tag{3.8.22}$$

The last bound is given by the mean value theorem and Hessian spectral

bound for N_{X_k} from assumption A1 as in Lemma 3.1:

$$\begin{aligned}\mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k)\|] &= \mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla \overline{F}_{X_k}(w^*) + \nabla \overline{F}_{X_k}(w^*)\|] \\ &\leq \mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w_k) - \nabla \overline{F}_{X_k}(w^*)\|] + \mathbb{E}_k[\|\nabla \overline{F}_{X_k}(w^*)\|] \\ &\leq L_{N_{X_k}}\|w_k - w^*\| + \epsilon_g. \end{aligned} \tag{3.8.23}$$

For GMRES, due to Proposition 6.33 in [114]

$$\mathcal{E} = \frac{L_{N_{S_k}}}{\gamma - \epsilon_H}\frac{C_m(\frac{a}{d})}{|C_m(\frac{c}{d})|}. \tag{3.8.24}$$

For MINRES, due to Theorem 5.10 in [114]

$$\mathcal{E} = \left(1 - \frac{(\gamma - \epsilon_H)^2}{L^2_{N_{S_k}}}\right)^{\frac{k}{2}}. \tag{3.8.25}$$

□

The constant c_0 will be small when the Monte Carlo error for the gradient is small, and the approximation of the linear solve via GMRES is accurate. The constant c_1 will be small when the full Newton step $\alpha_k = 1$ can be taken, the Monte Carlo error for the Hessian is small, and the approximation of the linear solve via GMRES is accurate. The constant c_2 will be small when the Hessian is well conditioned.

Remark. *In Theorem 3.6 the constant c_0 depends not only on the Monte Carlo approximation of the gradient, but also on the error in the Krylov solve $\mathcal{E}$ and the constant ϵ_g from Assumption A5. In order to derive a linear-quadratic convergence rate from this bound in the semi-stochastic case, one needs to employ the restrictive assumption that $\epsilon_g = 0$, i.e. w^* is a local minimum for all of the sample gradients.*

3.8.3 Convergence of Low Rank Newton Method

Before I state a bound for the convergence in expected value of stochastic low rank Newton methods based on SVD, I first have some lemmas.

Lemma 3.3. *Let $\{w_k\}$ be the iterates generated by (3.5.12), and suppose that assumptions A1-A3 hold, then for each k*

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq \frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\Bigg[\frac{M}{2}\|w_k - w^*\|^2 + L_{N_{S_k}}|1 - \alpha_k|\|w_k - w^*\| + \mathbb{E}_k\|([H_k^{(r)} + \gamma I](w_k) - \nabla^2\overline{F}(w_k))(w_k - w^*)\| + \frac{\alpha_k v}{\sqrt{N_{X_k}}}\Bigg], \tag{3.8.26}$$

where $\overline{\lambda_r^{(k)}} = \mathbb{E}_k[\lambda_r^{(k)}]$.

Proof. Expanding the left hand side of equation (3.8.26) and using the triangle inequality one can derive the following bound:

$$\begin{aligned}
&\mathbb{E}_k[\|w_{k+1} - w^*\|] \\
&= \mathbb{E}_k[\|w_k - w^* - \alpha_k[H_k^{(r)} + \gamma I]^{-1}(w_k)\nabla\overline{F}_{X_k}(w_k)\|] \\
&= \mathbb{E}_k[\|[H_k^{(r)} + \gamma I]^{-1}([H_k^{(r)} + \gamma I](w_k)(w_k - w^*) \\
&\quad - \alpha_k\nabla\overline{F}(w_k) - \alpha_k\nabla\overline{F}_{X_k}(w_k) + \alpha_k\nabla\overline{F}(w_k))\|] \\
&\leq \frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\mathbb{E}_k[\|\big([H_k^{(r)} + \gamma I](w_k) - \nabla^2\overline{F}(w_k)\big)(w_k - w^*) \\
&\qquad\qquad + \nabla^2\overline{F}(w_k)(w_k - w^*) - \alpha_k\nabla\overline{F}(w_k)\|] \\
&\quad + \frac{\alpha_k}{|\overline{\lambda_r^{(k)}} + \gamma|}\mathbb{E}_k[\|\nabla\overline{F}_{X_k}(w_k) - \nabla\overline{F}(w_k)\|].
\end{aligned} \tag{3.8.27}$$

Therefore,

$$\begin{aligned}
\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq\ & \underbrace{\frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\|\nabla^2\overline{F}(w_k)(w_k - w^*) - \alpha_k\nabla\overline{F}(w_k)\|}_{\text{term 1}} \\
&+ \underbrace{\frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\mathbb{E}_k[\|[H_k^{(r)} + \gamma I] - \nabla^2\overline{F}(w_k))(w_k - w^*)]}_{\text{term 2}} \\
&+ \underbrace{\frac{\alpha_k}{|\overline{\lambda_r^{(k)}} + \gamma|}\mathbb{E}_k[\|\nabla\overline{F}_{X_k}(w_k) - \nabla\overline{F}(w_k)\|]}_{\text{term 3}}.
\end{aligned} \tag{3.8.28}$$

For term 1, a bound is given by Lemma 3.1

$$\frac{1}{|\overline{\lambda_r^{(k)}}+\gamma|}\|\nabla^2\overline{F}(w_k)(w_k-w^*)-\nabla\overline{F}(w_k)\|$$
$$\leq \frac{M}{2|\overline{\lambda_r^{(k)}}+\gamma|}\|w_k-w^*\|^2+\frac{L_{N_{S_k}}|1-\alpha_k|}{|\overline{\lambda_r^{(k)}}+\gamma|}\|w_k-w^*\|. \tag{3.8.29}$$

Term 3 can be bounded by a result given in Lemma 2.2 of [15]:

$$\mathbb{E}_k[\|\nabla\overline{F}(w_k)-\nabla\overline{F}_{X_k}\|]\leq\frac{v}{\sqrt{N_{X_k}}}. \tag{3.8.30}$$

□

Lemma 3.4. *Bounds for exact truncated Hessian approximation. Suppose that Assumptions A1 and A4 hold, then*

$$\mathbb{E}_k\|[H_k^{(r)}+\gamma I](w_k)-\nabla^2\overline{F}(w_k))(w_k-w^*)\|\leq\left[|\overline{\lambda_{r+1}^{(k)}}|+\gamma+\frac{\sigma}{\sqrt{N_{S_k}}}\right]\|w_k-w^*\|. \tag{3.8.31}$$

Proof. Expanding the left hand side of equation (3.8.31), and employing the triangle inequality, the following bound can be derived:

$$\mathbb{E}_k\|([H_k^{(r)}(w_k)+\gamma I]-\nabla^2\overline{F}(w_k))(w_k-w^*)\|\leq$$
$$\underbrace{\mathbb{E}_k\|([H_k^{(r)}(w_k)+\gamma I]-\nabla^2\overline{F}_{S_k}(w_k))(w_k-w^*)\|}_{\text{term 1}} \tag{3.8.32}$$
$$+\underbrace{\mathbb{E}_k\|(\nabla^2\overline{F}(w_k)-\nabla^2\overline{F}_{S_k}(w_k))(w_k-w^*)\|}_{\text{term 2}}. \tag{3.8.33}$$

The first term is bounded as

$$\mathbb{E}_k\|([H_k^{(r)}(w_k)-\nabla^2\overline{F}_{S_k}(w_k))+\gamma I](w_k-w^*)\|$$
$$\leq\mathbb{E}_k\|(H_k^{(r)}(w_k)-\nabla^2\overline{F}_{S_k}(w_k))(w_k-w^*)\|+\gamma\|w_k-w^*\| \tag{3.8.34}$$
$$\leq\mathbb{E}_k\|H_k^{(r)}(w_k)-\nabla^2\overline{F}_{S_k}(w_k)\|\|(w_k-w^*)\|+\gamma\|w_k-w^*\|$$
$$=(|\overline{\lambda_{r+1}^{(k)}}|+\gamma)\|w_k-w^*\|. \tag{3.8.35}$$

The second term is bounded by

$$\frac{\sigma}{\sqrt{N_{S_k}}}\|w_k - w^*\|, \tag{3.8.36}$$

via Lemma 2.3 in [15]. □

Lemma 3.5. *Bounds for randomized truncated Hessian approximation. Suppose that assumptions A1 and A4 hold, and* $2 \leq r \leq \frac{d}{2}$*, and* $H_k^{(r)}$ *is calculated via randomized SVD, with random matrices drawn from a Gaussian probability measure* ρ*. Then:*

$$\begin{aligned}
&\mathbb{E}_k\big[\mathbb{E}_\rho\|[H_k^{(r)} + \gamma I](w_k) - \nabla^2\overline{F}(w_k))(w_k - w^*)\|\big] \\
&\leq \left(\left(1 + 4\frac{\sqrt{d(r+p)}}{p-1}\right)\overline{|\lambda_{r+1}^{(k)}|} + \gamma + \frac{\sigma}{\sqrt{N_{S_k}}}\right)\|w_k - w^*\|
\end{aligned} \tag{3.8.37}$$

Proof. Expanding the left hand side of equation (3.8.37) and employing the triangle inequality, the following bound can be established:

$$\begin{aligned}
&\mathbb{E}_k\big[\mathbb{E}_\rho\|([H_k^{(r)}(w_k) + \gamma I] - \nabla^2\overline{F}(w_k))(w_k - w^*)\|\big] \leq \\
&\underbrace{\mathbb{E}_k\big[\mathbb{E}_\rho\|([H_k^{(r)}(w_k) + \gamma I] - \nabla^2\overline{F}_{S_k}(w_k))(w_k - w^*)\|\big]}_{\text{term 1}} &(3.8.38)\\
&\qquad + \underbrace{\mathbb{E}_k\|(\nabla^2\overline{F}(w_k) - \nabla^2\overline{F}_{S_k}(w_k))(w_k - w^*)\|}_{\text{term 2}} &(3.8.39)
\end{aligned}$$

The first term is bounded as

$$\begin{aligned}
&\mathbb{E}_k\big[\mathbb{E}_\rho\|([H_k^{(r)}(w_k) - \nabla^2\overline{F}_{S_k}(w_k)) + \gamma I](w_k - w^*)\|\big] \\
&\leq \mathbb{E}_k\big[\mathbb{E}_\rho\|(H_k^{(r)}(w_k) - \nabla^2\overline{F}_{S_k}(w_k))(w_k - w^*)\|\big] + \gamma\|w_k - w^*\| &(3.8.40)\\
&\leq \mathbb{E}_k\big[\mathbb{E}_\rho\|H_k^{(r)}(w_k) - \nabla^2\overline{F}_{S_k}(w_k)\|\|(w_k - w^*)\|\big] + \gamma\|w_k - w^*\| \\
&\leq \mathbb{E}_k\left(\left(1 + 4\frac{\sqrt{d(r+p)}}{p-1}\right)|\lambda_{r+1}^{(k)}| + \gamma\right)\|w_k - w^*\| \\
&= \left(\left(1 + 4\frac{\sqrt{d(r+p)}}{p-1}\right)\overline{|\lambda_{r+1}^{(k)}|} + \gamma\right)\|w_k - w^*\|, &(3.8.41)
\end{aligned}$$

where the second to last bound comes from equation 1.8 in [58]. The second term is bounded by

$$\frac{\sigma}{\sqrt{N_{S_k}}}\|w_k - w^*\|, \tag{3.8.42}$$

via Lemma 2.3 in [15]. □

Now I can state a convergence rate for the stochastic low rank Newton method.

Theorem 3.5 (Local convergence of stochastic low rank Newton)**.** *Let* $\{w_k\}$ *be the iterates generated by* (3.5.12)*, let* w^* *be a stationary point and suppose that assumptions A1 - A4 hold, then for each* k

$$\mathbb{E}_k[\|w_{k+1} - w^*\|] \leq c_0 + c_1\|w_k - w^*\| + c_2\|w_k - w^*\|^2, \tag{3.5.13}$$

where

$$c_0 = \frac{\alpha_k v}{|\overline{\lambda_r^{(k)}} + \gamma|\sqrt{N_{X_k}}}, \tag{3.5.14a}$$

$$c_1 = \frac{1}{|\overline{\lambda_r^{(k)}} + \gamma|}\left[L_{N_{S_k}}|1 - \alpha_k| + \mathcal{E}|\overline{\lambda_{r+1}^{(k)}}| + \gamma + \frac{\sigma}{\sqrt{N_{S_k}}}\right], \tag{3.5.14b}$$

$$c_2 = \frac{M}{2|\overline{\lambda_r^{(k)}} + \gamma|}. \tag{3.5.14c}$$

Here I define $\overline{\lambda_r^{(k)}} = \mathbb{E}_k[\lambda_r^{(k)}]$*. The error coefficient* $\mathcal{E} = 1$ *when the truncated low rank spectral decomposition is exact, and* $\mathcal{E} = \left(1 + 4\frac{\sqrt{d(r+p)}}{p-1}\right)$ *in the case that it is calculated using randomized SVD.*

Proof. This result follows immediately from Lemma (3.3), Lemma (3.4) and Lemma (3.5). □

Chapter 4

Conclusions

The main contributions of this thesis are related to two main lines of inquiry:

1. How to efficiently parametrize neural network surrogates for parametric maps defined by PDEs, so that they are dimension independent and can be trained with small training data sets.

2. How to efficiently use second order information to improve solutions to the related neural network training problem.

The first line of inquiry is motivated by outer loop / many-query applications that involve expensive models parametrized by high dimensional nonlinear PDEs. The data-driven approach to surrogate construction that has been successful in typical machine learning applications is not suitable for high dimensional nonlinear parametric mappings involving nonlinear or otherwise expensive-to-solve PDEs. In this setting data generation for training is very expensive, since each training data pair requires a query of an expensive nonlinear model; in some settings individual evaluations can take days to weeks. Further since data come from high dimensional PDE based models, the dimensionality of the outputs can be very large. In this setting, data-driven approaches lead to very high dimensional configuration spaces for surrogate models, with very few training data to infer the surrogate weights.

In this work I investigate the use of model information to detect informed subspaces of the input and output where the parametric map can be represented efficiently. This leads to a discretization dimension independent approach for parametric map surrogate construction. The model is

parametrized by the intrinsic dimensionality of the mapping, instead of the discretization dimension which may be several orders of magnitude larger.

This framework helps identify key dimensionality of the parametric mapping, and builds parsimonious surrogates that only attempt to resolve the mapping in these informed subspaces. Data-driven approaches try to resolve the mapping in the whole input and output spaces. This drastically increases the amount of training data required to infer the weights in the surrogate model. Since the complementary subspaces are where the input-output mapping is not informed, this can lead to learning stochastic variations in the training data that are not representative of the true mapping in these complementary subspaces. This can lead to overfitting and poor generalizability.

This approach suggests that number of training data needed for surrogate construction need only scale linearly with the intrinsic dimensionality, thus allowing a principled way to generate small data sets and train a model in informed modes of the input-output mapping. Numerical experiments suggest that identification of intrinsic dimensionality is key to achieving good performance of surrogate models. Models parametrized by dominant subspaces of the input and output performed as well as data-driven approaches, for a fraction of the configuration space dimension, when the discretization dimension was not too large. As the discretization dimension increased, the parsimonious dimension independent strategy continued to perform well and the data-driven approach performed worse and worse.

The general approach demonstrates that restricting neural network representations to only dominant subspaces of the input and output is a principled strategy for model based parametric maps. In future work, I hope to spend more time studying more neural network parametrizations and refining the approach (e.g. different neural network structures, initializations, hyperparameters).

This work was restricted to moderate sized problems that could generate datasets using single core computers. In future work, I hope to extend

these approximations to very high dimensional mappings coming from very large models. Specifically I plan to test these approaches on ocean dynamics applications (MITgcm), large scale ice sheet applications, and aerodynamic wing design applications.

The second line of inquiry focused on how to use second order Hessian information to build efficient optimizers that lead to quality solutions. Major issues in this research inquiry are the computational cost of the method, indefiniteness, and saddle points, which can hobble the performance of optimizers, convergence properties, and generalization error.

I began by analyzing statistical properties of subsampled indefinite Hessians. A statistical model for the subsampled Hessian suggested that in regions of parameter space where eigenvalues of the true stochastic Hessian change sign, subsampled approximations could exhibit unbounded variance. This led to numerical experiments that studied the variance of approximations of eigenvalues of the true stochastic Hessian. These numerical experiments demonstrated that when the neural network training Hessian was indefinite, small eigenvalues of the Hessian could exhibit large degrees of variance. This means that the spectra of specific subsampled approximations of the stochastic Hessian could be dominated by noise in small eigenvalues. This motivates investigation of stochastic Newton methods based on what kind of spectral information they use in an approximate Newton solve. I analyze matrix-free Newton methods, paying attention to computational cost, convergence properties, and how they can be made robust to noise in the Hessian spectrum.

I first analyze inexact Newton Krylov methods. When the spectrum of the Hessian decays quickly or clusters, these methods require few matrix-vector products and can achieve fast convergence in the vicinity of local minima. These methods can be made somewhat robust to saddle points via early termination schemes, but unfortunately they attempt to resolve lower order noise dominated modes of the Hessian. This is problematic since it can

lead to overfitting, or rescale certain components of the gradient by noisy approximations of the Hessian eigenvalues and take iterates out of basins of attraction (this is seen in numerical results).

In order to only attempt to resolve persistent eigenvalues of the Hessian matrix, and to facilitate fast escape from indefinite regions / saddle points I propose the low rank saddle free Newton (LRSFN) algorithm. This algorithm is motivated by analysis and numerical experiments that demonstrate only dominant eigenvalues of the Hessian are not dominated by noise for stochastic indefinite Hessians. When the Hessian is low rank in the vicinity of a local minimum, LRSFN can achieve fast convergence.

I test these methods on some modest neural network training problems: MNIST classification problems, a CIFAR10 autoencoder problem, and a parametric map surrogate regression problem coming from the other part of my thesis. In these numerical experiments I demonstrate that LRSFN and INCG can be competitive with first order methods in terms of computational cost, and perform better in convergence and generalization. LRSFN and INCG performed the best on the examples that I investigated, required the least amount of hyperparameter tuning, and were not sensitive to the scaling of the problem as the first order methods were. LRSFN performed especially well in terms of generalization, it typically leading to the least amount of overfitting of all of the methods, and usually found the best generalizable solution. These Newton methods were especially useful in least squares type problems such as the autoencoder problem and parametric surrogate problem.

Matrix-free Newton methods should be considered mainstream algorithms in stochastic nonconvex optimization problems. They can efficiently exploit low dimensional geometric information about the problem to facilitate fast convergence, and they can be parametrized to only resolve persistent modes of the geometric information, which leads to better generalizability in addition to superior convergence.

In future work I hope to extend these methods to higher dimensional problems. The Hessian spectrum statistical experiments I did limited the problem sizes I considered substantially. Computing sample statistics of eigenvalues of subsampled Hessians at every iteration is extremely computationally expensive. It is not necessary for fast performance of the method, but was necessary for an understanding of the statistical properties of the Hessian during training. The Hessian approximation used in LRSFN allows for parallelization, which is a key issue for scaling optimization methods to large scale problems. Other matrix-free Newton methods typically are sequential by nature. Due to this computational concurrency, LRSFN is a practical optimizer at very large scales. In future work I plan on scaling the LRSFN method to much higher dimensional neural network training problems. In addition, I hope to look into efficient regularization techniques to address Hessian nullspaces with implicit constraints. This idea is explored briefly in Appendix 1.

In total I demonstrate that second order methods can exploit low dimensional geometry of the stochastic nonconvex optimization problem to achieve fast convergence and better generalization properties. In this work it is shown that indefiniteness, stochasticity, and noise in second order information are linked, and this should be considered for the designer of an efficient second order method.

The main theme of this thesis the development of scalable algorithms that use low dimensional structure to exploit robust information in high dimensional stochastic problems.

In the inquiry into efficient neural network architecture construction, this is done by identifying low dimensional subspaces of input and output spaces for parametric maps where the output is sensitive to the input parameter *in expectation*. Attempting to reconstruct relationships in these complementary modes may lead to learning stochastic variations in training data and thus overfitting. This low dimensional restriction serves as a regularizing con-

straint for the construction of the surrogate, as complementary modes are by definition not representable in this framework. This is especially important when one cannot generate extensive training datasets for the surrogate construction. When the generation of training data is expensive, one ought to prioritize resolving modes of the map that are important in expected value. This low dimensional structure also allows for a scalable approach, which is critical for high dimensional problems.

In the inquiry related to stochastic second order optimizers, analysis suggests that spectral properties of stochastic indefinite Hessians may only be not dominated by noise in low dimensional dominant subspaces. Attempting to resolve noisy lower order information can lead to poor performance and overfitting for an optimizer. The LRSFN algorithm only attempts to rescale the gradient direction using robust curvature information of the Hessian; this leads to an efficient algorithm with good performance. This algorithm is scalable due to its ability to handle computational concurrency in the Hessian approximation, which only resolves a low dimensional subspace. And finally, since the spectral modes in these subspaces do not exhibit high variance, fewer samples can be used to approximate these modes. This leads to an algorithm that is competitive with first order methods in terms of work, but can efficiently exploit dominant curvature information that makes the method less sensitive to the geometric scaling of the optimization problem than first order methods, and leads to better convergence properties and generalization.

Appendix

Appendix 1

Ill-Posedness of Neural Network Training

1.1 Introduction

Here, I characterize the optimization geometry of nonlinear least-squares regression problems for generic dense neural networks and analyze the ill-posedness of the training problem.[1] Neural networks are a popular nonlinear functional approximation technique that are successful in data driven approximation regimes. A one-layer neural network can approximate any continuous function on a compact set, to a desired accuracy given enough neurons [39, 63]. Dense neural networks have been shown to be able to approximate polynomials arbitrarily well given enough hidden layers [121]. While no general functional analytic approximation theory exists for neural networks, they are widely believed to have great approximation power for complicated patterns in data [106].

Training a neural network, i.e., determining optimal values of network parameters to fit given data, can be accomplished by solving the nonconvex optimization problem of minimizing a loss function (known as empirical risk minimization). Finding a global minimum is NP-hard and instead one usually settles for local minimizers [13, 93]. Here I seek to characterize how nonlinear activation functions affect the least-squares optimization geometry at stationary points. In particular, I wish to characterize the conditions for strict saddle points and spurious local minima. Strict saddle points are stationary points where the Hessian has at least one direction of strictly negative

[1] This chapter contains content from [98] (Thomas O'Leary-Roseberry, Omar Ghattas. Ill-Posedness and Optimization Geometry for Nonlinear Neural Network Training arXiv preprint, arxiv:2002.02882, 2020)

curvature. They do not pose a significant problem for neural network training, since they can be escaped efficiently with first and second order methods [41, 68, 69, 94, 96]. On the other hand, spurious local minima (where the gradient vanishes but the data misfit is nonzero) are more problematic; escaping from them in a systematic way may require third order information [8].

I also seek to analyze the rank deficiency of the Hessian of the loss function at global minima (if they exist), in order to characterize the ill-posedness of the nonlinear neural network training problem. Training a neural network is, mathematically, an inverse problem; rank deficiency of the Hessian often makes solution of the inverse problem unstable to perturbations in the data and leads to severe numerical difficulties when using finite precision arithmetic [60]. While early termination of optimization iterations often has a regularizing effect [59, 44], and general-purpose regularization operators (such as ℓ^2 or ℓ^1) can be invoked, when to terminate the iterations and how to choose the regularization to limit bias in the solution are omnipresent challenges. On the other hand, characterizing the nullspace of the Hessian can provide a basis for developing a principled regularization operator that parsimoniously annihilates this nullspace, as has been recently done for shallow linear neural networks [140].

I consider both shallow and deep dense neural network parametrizations. The dense parametrization is sufficiently general since convolution operations can be represented as cyclic matrices with repeating block structure. For the sake of brevity, I do not consider affine transformations, but this work can easily be extended to this setting. I begin by analyzing shallow dense nonlinear networks, for which we show that the nonlinear activation function plays a critical role in classifying stationary points. In particular, if the neural network can exactly fit the data, and zero misfit global minima exist, I show how the Hessian nullspace depends on the activation function and its first derivative at these points.

For linear networks, results about local minima, global minima, strict saddle points, and optimal regularization operators have been shown [9, 140]. The linear network case is a nonlinear matrix factorization problem, given data matrices $X \in \mathbb{R}^{n\times d}, Y \in \mathbb{R}^{m\times d}$, one seeks to find $W_1^* \in \mathbb{R}^{m\times r}, W_0^* \in \mathbb{R}^{r\times n}$ such that they minimize

$$\frac{1}{2}\|Y - W_1 W_0 X\|_F^2. \tag{1.1.1}$$

When the data matrix X has full row rank, then by the Eckart-Young Theorem, the solution is given by the rank r SVD of $YX^T(XX^T)^{-1}$, which is denote with a subscript r

$$W_1^* W_0^* = [YX^T(XX^T)^{-1}]_r. \tag{1.1.2}$$

The solution is non-unique since for any invertible matrix $B \in \mathbb{R}^{r\times r}$

$$(W_1^* B)(B^{-1}W_0^*) = [YX^T(XX^T)^{-1}]_r \tag{1.1.3}$$

is also a solution. I show that in addition to inheriting issues related to ill-posedness of matrix factorization, the nonlinear activation functions in the nonlinear training problem create ill-posedness and non-uniqueness.

Stationary points not corresponding to zero misfit global minima are determined by the activation function and its first derivative through an orthogonality condition. In contrast to linear networks, for which the existence of spurious local minima depends only on the rank of the training data and the weights. For nonlinear networks, both spurious local minima and strict saddle points exist, and depend on the activation functions, the training data, and the weights.

I extend these results to deep dense neural networks where stationary points can arise from exact reconstruction of the training data by the network, or an orthogonality condition that involves the activation functions of each layer of the network and their first derivatives.

For nonlinear neural networks, some work exists on analyzing networks with ReLU activation functions; in particular Safran et. al. establish conditions for the existence of spurious local minima for two layer ReLU networks [116].

1.1.1 Notation and Definitions

For a given matrix $A \in \mathbb{R}^{m\times n}$, its vectorization, $\text{vec}(A) \in \mathbb{R}^{mn}$ is an mn vector that is the columns of A stacked sequentially. Given a vector $z \in \mathbb{R}^m$, its diagonalization $\text{diag}(z) \in \mathbb{R}^{m\times m}$ is the diagonal matrix with entry ii being component i from z. The diagvec operation is the composition $\text{diagvec}(A) = \text{diag}(\text{vec})(A) \in \mathbb{R}^{mn\times mn}$, this is sometimes shortened to dvec. The identity matrix in $\mathbb{R}^{d\times d}$ is denoted I_d. The notation $\nabla_X f(X)$ is used to mean derivatives of a function f with respect to a matrix X, and $\partial_{\text{vec}(X)} f(\text{vec}(X))$ when expressing derivatives with respect to a vectorized matrix $\text{vec}(X)$: $\partial_{\text{vec}(X)} f(X) = \frac{\partial f}{\partial \text{vec}(X)}$. For matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \text{ı} \times \text{ıı}$, the Kronecker product $A \otimes B \in \mathbb{R}^{pm\times qn}$ is the block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \tag{1.1.4}$$

For matrices $A, B \in \mathbb{R}^{m\times n}$, $A \circ B \in \mathbb{R}^{m\times n}$ is the Hadamard (element-wise) product. For a matrix $A \in \mathbb{R}^{m\times n}$ and a matrix $B \in \mathbb{R}^{n\times k}$, the expression $A \perp B$ means that the rows of A are orthogonal to the columns of B, and thus $AB = 0$.

For a differentiable function $F : \mathbb{R}^{d_W} \to \mathbb{R}$, and a parameter $W_0 \in \mathbb{R}^{d_W}$, W_0 is a first order stationary point if $\nabla F(W_0) = 0$, W_0 is a strict saddle point if there exists a negative eigenvalue for the Hessian $\nabla^2 F$. The matrix W_0 is a local minimum if the eigenvalues of the Hessian $\nabla^2 F$ are all nonnegative. The matrix W_0 is a global minimum if $F(W_0) \leq F(W)$ for all $W \in \mathbb{R}^{d_W}$.

1.2 Stationary Points of Shallow Dense Network

Consider a one layer dense neural network training problem. Given training data matrices $X \in \mathbb{R}^{n\times d}, Y \in \mathbb{R}^{m\times d}$, the shallow neural network architecture consists of an encoder weight matrix $W_0 \in \mathbb{R}^{r\times n}$, a nonlinear activation function σ (which is applied element-wise), and then a decoder weight matrix $W_1 \in \mathbb{R}^{m\times r}$. The training problem (empirical risk minimization) may then be stated as

$$\begin{aligned}\min_{W_1,W_0} F(W_1,W_0) &= \sum_{i=1}^{d} \frac{1}{2}\|y_i - W_1\sigma(W_0x_i)\|^2_{\ell^2(\mathbb{R}^m)} \\ &= \frac{1}{2}\|Y - W_1\sigma(W_0X)\|^2_{F(\mathbb{R}^{m\times d})}. \end{aligned} \tag{1.2.1}$$

I begin by analyzing first order stationary points of the objective function F.

Theorem 1.1. *The gradient of the objective function F is given by*

$$\begin{aligned}\nabla F(W_1,W_0) &= [\nabla_{W_1}F(D,E)^T, \nabla_{W_0}F(W_1,W_0)^T]^T \\ \nabla_{W_1}F(W_1,W_0) &= (W_1\sigma(W_0X) - Y)\sigma(W_0X)^T \qquad (1.2.2) \\ \nabla_{W_0}F(W_1,W_0) &= \\ &\quad [\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))]X^T. \qquad (1.2.3)\end{aligned}$$

First order stationary points are characterized by two main conditions:

1. *A global minimum where the misfit is exactly zero: $W_1\sigma(W_0X) = Y$. The possibility for which depends on the representation capability of the network, and the data.*

2. *A stationary point not corresponding to zero misfit: $\sigma'(W_0X)\circ W_1^T(W_1\sigma(W_0X) - Y) \perp X^T$, and $(W_1\sigma(W_0X) - Y) \perp \sigma(W_0X)^T$*

Proof. The partial derivatives of the objective function $F(W_1,W_0)$ are derived in Lemma 1.2. At a first order stationary point of the objective function

F both partial derivatives must be zero:

$$(W_1\sigma(W_0X) - Y)\sigma(W_0X)^T = 0 \tag{1.2.4}$$

$$[\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))]X^T = 0. \tag{1.2.5}$$

In the case that $W_1\sigma(W_0X) = Y$ then both terms are zero, and the corresponding choices of W_1, W_0 define a global minimum. This can be seen since F is a nonnegative function, and in this case it is exactly zero.

Stationary points where $W_1\sigma(W_0X) \neq Y$ are characterized by orthogonality conditions. If $\nabla_{W_1}F(W_1, W_0) = 0$, then this means that $(W_1\sigma(W_0X) - Y) \perp \sigma(W_0X)^T$; that is, the rows of $W_1\sigma(W_0X) - Y$ and the columns of $\sigma(W_0X)^T$ are pairwise orthogonal. If $\nabla_{W_0}F(W_1, W_0) = 0$, then this similarly means that $[\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))] \perp X^T$ □

Corollary 1.1. *Any W_1, W_0 such that $\sigma(W_0X) = 0$ and $\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y)) = 0$ correspond to first order stationary points of the objective function F. In particular, any W_0 for which $\sigma(W_0X) = \sigma'(W_0X) = 0$ corresponds to a first order stationary point for all W_1.*

This result implies that points in parameter space where the activation function and its derivatives are zero can lead to sub-optimal stationary points. Note that if a zero misfit minimum is not possible, there may or may not be an actual global minimum (there will always be a global infimum), but since the misfit is not zero any such point will still fall into the second category. In what follows I characterize the optimization geometry of the objective function F at global minima, and degenerate points of the activation function, i.e. points for which $\sigma(W_0X) = \sigma'(W_0X) = 0$.

1.2.1 Zero misfit minima

Suppose that for given data X, Y, there exists W_1, W_0, σ such that $W_1\sigma(W_0X) = Y$. As was discussed in Theorem 1.1, such points correspond to a global minimum. In what follows I characterize Hessian nullspace at these points, and corresponding ill-posedness of the training problem.

Theorem 1.2. *Characterization of Hessian nullspace at global minimum. Given data X, Y, σ suppose there exist weight matrices W_1, W_0 such that $Y = W_1\sigma(W_0X)$. Suppose further that W_1 and $\sigma(W_0X)$ are full rank, then the Hessian nullspace is characterized by directions $\widehat{W_1}, \widehat{W_0}$ such that*

$$\left[\widehat{W_1}\sigma(W_0X) + W_1(\sigma'(W_0X) \circ \widehat{W_0}X)\right] \perp \sigma(W_0X)^T \tag{1.2.6}$$

$$\left[[\sigma'(W_0X) \circ (W_1^T\widehat{W_1}\sigma(W_0X))]+ \right.$$

$$\left.[\sigma(W_0X) \circ (W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X))]\right] \perp X^T. \tag{1.2.7}$$

In particular for any direction $\widehat{W_0}$, such that the directional derivative $\sigma'(W_0X) \circ \widehat{W_0}X$ is zero, the weight matrices

$$\widehat{W_0}$$

$$\widehat{W_1} = -W_1(\sigma'(W_0X) \circ \widehat{W_0}X)\sigma(W_0X)^T[\sigma(W_0X)\sigma(W_0X)^T]^{-1} \tag{1.2.8}$$

are in the nullspace of the Hessian matrix $\nabla^2 F(W_1, W_0)$

Proof. Since the misfit is zero, the Hessian is exactly the Gauss-Newton Hessian, which is derived in Lemma 1.3. The matrices $\widehat{W_0}, \widehat{W_1}$ are in the nullspace of the Hessian if

$$\left[\widehat{W_1}\sigma(W_0X) + W_1(\sigma'(W_0X) \circ \widehat{W_0}X)\right]\sigma(W_0X)^T = 0 \tag{1.2.9}$$

$$\left[[\sigma'(W_0X) \circ (W_1^T\widehat{W_1}\sigma(W_0X))]+ \right.$$

$$\left.[\sigma(W_0X) \circ (W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X))]\right]X^T = 0. \tag{1.2.10}$$

For this to be the case, it must be the case that

$$[\widehat{W_1}\sigma(W_0X) + W_1(\sigma'(W_0X) \circ \widehat{W_0}X)] \perp \sigma(W_0X)^T \tag{1.2.11}$$

and

$$[[\sigma'(W_0X) \circ (W_1^T\widehat{W_1}\sigma(W_0X))]+[\sigma(W_0X) \circ (W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X))]] \perp X^T. \tag{1.2.12}$$

The Hessian nullspace is fully characterized by points $\widehat{W_1}, \widehat{W_0}$ that satisfy these two orthogonality constraints. One way in which these constraints are satisfied is if

$$\widehat{W_1}\sigma(W_0X) = -W_1(\sigma'(W_0X) \circ \widehat{W_0}X) \tag{1.2.13}$$

$$[\sigma'(W_0X) \circ (W_1^T\widehat{W_1}\sigma(W_0X))]+$$

$$[\sigma(W_0X) \circ (W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X))] = 0. \tag{1.2.14}$$

Subsituting (1.2.13) into (1.2.14) one can obtain

$$[-\sigma'(W_0X) + \sigma(W_0X)] \circ (W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X)) = 0. \tag{1.2.15}$$

The first term is nonzero if $\sigma \neq \exp$, since $\sigma(W_0X)$ is assumed to be full rank. For the Hadamard product to be zero, the second term must be zero:

$$W_1^TW_1(\sigma'(W_0X) \circ \widehat{W_0}X) = 0. \tag{1.2.16}$$

This is accomplished when $W_1^TW_1 \perp \sigma'(W_0X) \circ \widehat{W_0}X$. Since W_1 is full rank this condition reduces to $\sigma'(W_0X) \circ \widehat{W_0}X = 0$. Suppose that $\widehat{W_0}$ satisfies this directional derivative constraint, then one can find a corresponding $\widehat{W_1}$ such that $\widehat{W_1}, \widehat{W_0}$ are in the Hessian nullspace from (1.2.13):

$$\widehat{W_1} =$$

$$-W_1(\sigma'(W_0X) \circ \widehat{W_0}X)\sigma(W_0X)^T[\sigma(W_0X)\sigma(W_0X)^T]^{-1}. \tag{1.2.17}$$

Note that $\sigma(W_0X)\sigma(W_0X)^T \in \mathbb{R}^{r\times r}$ is invertible since $\sigma(W_0X)$ is assumed to be full rank.

□

This result shows that the Hessian may have a nontrivial nullspace at zero misfit global minima; in particular, if there are any local directions $\widehat{W_0}$ satisfying the directional derivative constraint $\sigma'(W_0X) \circ \widehat{W_0}X = 0$, then the Hessian is guaranteed to have at least one zero eigenvalue. If the Hessian has at least one zero eigenvalue, then the candidate global minimum W_1, W_0 is

not unique, and instead is on a manifold of global minima. Global minima are in this case weak minima.

This result is similar to the non-uniqueness of the linear network training problem, Equation (1.1.3). However in this case the linear rank constraints are obfuscated by the nonlinear activation function, and, additionally the zeros of the activation function lead to more possibility for Hessian rank-deficiency and associated ill-posedness.

For weak global minima, regularization schemes that annihilate the Hessian nullspace while leaving the range space unscathed can be used to make the training problem well-posed without biasing the solution. Furthermore, such regularization schemes will accelerate the asymptotic convergence rates of second order methods (Newton convergence deteriorates from quadratic to linear in the presence of singular Hessians), thereby making them even more attractive relative to first order methods.

1.2.2 Strict Saddle Points and Spurious Local Minima.

As was shown in Theorem 1.1 and Corollary 1.1, there are stationary points where the misfits are not zero. In this section it is demonstrated that these points can be both strict saddle points as well as spurious local minima.

Suppose the gradient is zero, but the misfit is nonzero. As was discussed in condition 2 of Theorem 1.1 such minima require orthogonality conditions for matrices that show up in the gradient. Corollary 1.1 establishes that this result is achieved if $\sigma(W_0X) = \sigma'(W_0X) = 0$. Many activation functions such as ReLU, sigmoid, softmax, softplus, tanh have many points satisfying these conditions (or at least approximately satisfying these conditions, i.e. for small $\epsilon > 0$, $\|\sigma'(W_0X)\|_F, \|\sigma(W_0X)\|_F \leq \epsilon$). Such stationary points are degenerate due to the activation functions. In what follows I show that while these points are likely to be strict saddles, it is possible that some of them have no directions of negative curvature and are thus spurious local minima.

Theorem 1.3. *Negative Curvature Directions at Degenerate Activation Sta-*

tionary Points. Let W_1 be arbitrary and suppose that W_0 is such that $\sigma'(W_0X) = \sigma(W_0X) = 0$, negative curvature directions of the Hessian at such points are characterized by directions $\widehat{W_0}$ such that

$$\sum_{k=1}^{d}\sum_{i=1}^{r}(\widehat{W_0}x^{(k)})_i^2(\sigma''(W_0x^{(k)}))_i(W_1^Ty^{(k)})_i < 0. \tag{1.2.18}$$

Proof. Since $\sigma'(W_0X)$ all of the terms in the Gauss-Newton Hessian are zero (see Lemma 1.3). Further, all of the off-diagonal non Gauss-Newton portions are also zero. In this case the only block of the Hessian that is nonzero is the non Gauss-Newton $W_0 - W_0$ block (see Lemma 1.4). I proceed by analyzing an un-normalized Rayleigh quotient for this block in an arbitrary direction $\widehat{W_0}$. From Equation 1.6.24 one can compute the quadratic form:

$$\begin{aligned}
&\mathrm{vec}(\widehat{W_0})^T(\partial_{\mathrm{vec}(W_0)}\partial_{\mathrm{vec}(W_0)}\mathrm{misfit})^T\mathrm{misfit}\mathrm{vec}(\widehat{W_0}) \\
=&\mathrm{vec}(\widehat{W_0}X)^T\mathrm{dvec}((W_1^T(W_1\sigma(W_0X) - Y)) \\
&\qquad \circ \sigma''(W_0X))\mathrm{vec}(\widehat{W_0}X) \\
=&\mathrm{vec}(\widehat{W_0}X)^T\mathrm{vec}(\big[(W_1^T(W_1\sigma(W_0X) - Y)) \\
&\qquad \circ \sigma''(W_0X) \circ \widehat{W_0}X\big])
\end{aligned} \tag{1.2.19}$$

Expanding this term in a sum one can obtain

$$\sum_{k=1}^{d}\sum_{i=1}^{r}(\widehat{W_0}x^{(k)})_i^2(\sigma''(W_0x^{(k)}))_i(W_1^T(W_1\sigma(W_0x^{(k)}) - y^{(k)})_i \tag{1.2.20}$$

The result follows noting that $\sigma(W_0X) = 0$. □

Directions $\widehat{W_0}$ that satisfy the negative curvature condition (1.2.18) are difficult to understand in their generality, since they depend on X, Y and σ''. I discuss some example sufficient conditions.

Corollary 1.1. *Saddle point with respect to one data pair. Given W_1, W_0, and a strictly convex activation function σ, suppose that $\sigma'(W_0X) = \sigma(W_0X) = 0$. Suppose that there is a data pair with $x^{(k)} \neq 0$ such that at least one negative component of $W_1^Ty^{(k)}$. Then W_1, W_0 is a strict saddle point.*

Proof. If $(W_1^T y^{(k)})_i < 0$, and the $x_j^{(k)} \neq 0$, then the direction $\widehat{W}_{0ij} = 1$ with all other components zero defines a direction of negative curvature.

$$\begin{aligned}\sum_{i=1}^{r}&(\widehat{W}_0 x^{(k)})_i^2(\sigma''(W_0 x^{(k)}))_i(W_1^T y^{(k)})_i \\ &=(x^{(k)})_j^2(\sigma''(W_0 x^{(k)}))_j(W_1^T y^{(k)})_j < 0. \end{aligned} \tag{1.2.21}$$

□

Corollary 1.2. *Given W_1, W_0 and a strictly convex function σ and all elements of one row of $W_1^T Y$ are negative then W_1, W_0 is a strict saddle point.*

Proof. Let the i^{th} row of $W_1^T Y$ satisfy this condition, then any choice of $\widehat{W}_0 \neq 0$ such that all rows other than i are zero will define a direction of negative curvature. □

These conditions are rather restrictive, but demonstrate the nature of existence of negative curvature directions. As was stated before, the most general condition for a strict saddle is the existence of $\widehat{W}_0$ that satisfies Equation (1.2.18). I conjecture that such an inequality shouldn't be hard to satisfy, but as it is a nonlinear inequality finding general conditions for the existence of such $\widehat{E}$ is difficult. I have the following result about how the zeroes of the activation function and its derivatives can lead to spurious local minima.

Corollary 1.3. *For a given W_0, if $\sigma(W_0 X) = \sigma'(W_0 X) = \sigma''(W_0 X) = 0$, then the Hessian at this point is exactly zero and this point defines a spurious local minimum.*

Such points exist for functions like ReLU, sigmoid, softmax, softplus etc. Any activation function that has large regions where it is zero (or near zero) will have such points. The question is then, how common are they? For the aforementioned functions, the function and its derivatives are zero or near zero when the argument of the function is sufficiently negative. For these functions, and a given tolerance $\epsilon > 0$ there exists a constant $C \leq 0$ such

that for all $\xi < C$, $\sigma(\xi) \leq \epsilon$, $\sigma'(\xi) \leq \epsilon$ and $\sigma''(\xi) \leq \epsilon$. For ReLU (which does not have any derivatives at zero) $C = \epsilon = 0$. In one dimension this condition is true for roughly half of the real number line for each of these functions. For the condition to be true for a vector it must be true elementwise. So for the condition

$$\sigma(W_0 x^{(k)}) \leq \epsilon \text{ and } \sigma'(W_0 x^{(k)}) \leq \epsilon \text{ and } \sigma''(W_0 x^{(k)}) \leq \epsilon \tag{1.2.22}$$

to hold for a given input datum $x^{(k)}$; the encoder array must map each component of $x^{(k)}$ into the strictly negative orthant of $\mathbb{R}^r$. The probability of drawing a mean zero Gaussian random vector in $\mathbb{R}^r$ that is in the strictly negative orthant is 2^{-r}. Furthermore for this condition to hold for all of $W_0 X$ means it must be true for each column of the matrix $W_0 X$. The probability of drawing a mean zero Gaussian random matrix in $\mathbb{R}^{r \times d}$ such that each column resides in the strictly negative orthant is 2^{-rd}. In practice the linearly encoded input data matrix $W_0 X$ is unlikely to have the statistical properties of a mean zero Gaussian, but this heuristic demonstrates that these degenerate points may be improbable to encounter. If the Hessian is exactly zero, one needs third order information to move in a descent direction [8].

1.3 Extension to Deep Networks

In this section I briefly discuss the general conditions for stationary points of a dense neural network. I consider the parameterization. In this case the weights for an N layer network are $[W_0, W_1, \cdots, W_N]$, where $W_0 \in \mathbb{R}^{r_0 \times m}$, $W_N \in \mathbb{R}^{n \times r_{N-1}}$, and all other $W_j \in \mathbb{R}^{r_1 \times r_0}$. The activation functions σ_j are arbitrary. The network parameterization is

$$W_N \sigma_N(W_{N-1} \sigma_{N_1}(\cdots \sigma_1(W_0 X) \cdots)). \tag{1.3.1}$$

I have the following general result about first order stationary points of deep neural networks.

Theorem 1.4. *Stationary points of deep dense neural networks The blocks of the gradient of the least squares loss function for the deep neural network (Equation* (1.3.1)*) are as follows:*

$$\begin{aligned}\nabla_{W_j} F(\mathbf{W}) = \Big[&\sigma'_{j+1}(W_j\sigma_j\cdots\sigma_1(W_0X)\cdots)\circ\\ &\big(W_{j+1}^T\big(\sigma'_{j+2}(W_{j+1}\sigma_j\cdots\sigma_1(W_0X)\cdots)\circ\\ &\cdots\circ\big(W_{N-1}^T\big(\sigma'_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)\big)\cdots\\ &\circ\big(W_N^T(W_N\sigma_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)-Y)\big)\cdots\big)\big)\Big]\\ &\sigma_j(W_j\cdots\sigma_1(W_0X))^T. \end{aligned} \tag{1.3.2}$$

Stationary points of the loss function are characterized by two main cases:

1. *The misfit is exactly zero. If such points are possible, then these points correspond to local minima*

2. *For each block the following orthogonality condition holds:*

$$\begin{aligned}&\Big[\sigma'_{j+1}(W_j\sigma_j\cdots\sigma_1(W_0X)\cdots)\circ\\ &\big(W_{j+1}^T\big(\sigma'_{j+2}(W_{j+1}\sigma_j\cdots\sigma_1(W_0X)\cdots)\circ\\ &\cdots\circ\big(W_{N-1}^T\big(\sigma'_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)\big)\cdots\\ &\circ\big(W_N^T(W_N\sigma_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)-Y)\big)\cdots\big)\big)\Big]\\ &\perp\sigma_j(W_j\cdots\sigma_1(W_0X))^T \end{aligned} \tag{1.3.3}$$

This result follows from Lemma 1.5. There are many different conditions on the weights and activation functions that will satisfy the orthogonality requirement in Equation (1.3.3). One specific example is analogous to the condition in Corollary 1.1.

Corollary 1.1. *Any weights* $[W_0,\ldots,W_{N-1}]$ *such that*

$$\sigma_N(W_{N-1}\sigma_{N-1}(\cdots\sigma_1(W_0X)\cdots)) = 0 \tag{1.3.4}$$

$$\sigma'_N(W_{N-1}\sigma_{N-1}(\cdots\sigma_1(W_0X)\cdots)) = 0 \tag{1.3.5}$$

correspond to a first order stationary point for any W_N*.*

This is the case since the term that is zero in Equation (1.3.4) shows up in the W_N block of the gradient, and the term that is zero in Equation (1.3.5) shows up in every other block of the gradient via an Hadamard product due to the chain rule.

Analysis similar to that in Section 1.2 can be carried out to establish conditions for Hessian rank deficiency at zero misfit minima and corresponding ill-posedness of the training problem in a neighborhood, as well as analysis that may establish conditions for saddle points and spurious local minima. Due to limited space I do not pursue such analyses, but expect similar results. Specifically the last activation function and its derivatives seem to be critical in understanding the characteristics of stationary points, both their existence and Hessian rank deficiency. If the successive layer mappings prior to the last layer map $W_{N-1}\sigma_{N-1}(\dots\sigma_1(W_0X))$ into the zero set of the last activation and its derivatives then I believe spurious local minima are possible.

1.4 Addressing ill-posedness with regularization

In this work some implicit relationships (equations (1.2.9) and (1.2.10)) are derived that characterize conditions for vectors to be in the Hessian nullspace. Generic relations for directions in the nullspace can be represented by the matrix

$$f(\mathbf{W}, X) = 0. \tag{1.4.1}$$

This is equivalent to the norm condition

$$\|f(\mathbf{W}, X)\| = 0. \tag{1.4.2}$$

In order to make the training problem well posed, one hopes to address the nullspace of the Hessian in order to make a minimum locally unique. One can penalize directions in the nullspace of the Hessian by imposing the following regularization, with regularization parameter β

$$\frac{\beta}{\|f(\mathbf{W}, X)\|^2}. \tag{1.4.3}$$

If one could fully represent directions of the entire nullspace of the Hessian in implicit relationships, then this regularization scheme could make the problem well posed. This is something I hope to look into in future research.

1.5 Conclusion

For dense nonlinear neural networks, I have derived expressions characterizing the nullspace of the Hessian in the vicinity of global minima. These can be used to design regularization operators that target the specific nature of ill-posedness of the training problem. When a candidate stationary point is a strict saddle, appropriately-designed optimization algorithms will escape it eventually (how fast they escape will depend on how negative the most negative eigenvalue of the Hessian is). The analysis in this work shows that when the gradient is small, it can be due to an accurate approximation of the mapping $X \mapsto Y$, or it can be due to the orthogonality condition, Equation (2). Spurious local minima can be identified easily, since $\|Y - W_1\sigma(W_0X)\|_F$ will be far from zero. Whether or not such points are strict saddles or local minima is harder to know specifically since this can depend on many different factors, such as the zeros of the activation function and its derivatives. Such points can be escaped quickly using Gaussian random noise [68]. When in the vicinity of a strict saddle point with a negative curvature direction that is large relative to other eigenvalues of the Hessian, randomized methods can be used to identify negative curvature directions and escape the saddle point at a cost of a small number of neural network evaluations [96].

1.6 Shallow Dense Neural Network Derivations

1.6.1 Derivation of gradient

Derivatives are taken in vectorized form. In order to simplify notation I use the following:

$$\begin{aligned} F(W_1, W_0) &= \frac{1}{2}\text{misfit}^T\text{misfit} \\ \text{misfit} &= \text{vec}(Y - W_1\sigma(W_0X)). \end{aligned} \tag{1.6.1}$$

In numerator layout partial differentials with respect to a vectorized matrix X are as follows:

$$\partial_{\mathrm{vec}(X)}\left(\frac{1}{2}\mathrm{misfit}^T\mathrm{misfit}\right) = (\partial_{\mathrm{vec}(X)}\mathrm{misfit})^T\mathrm{misfit}. \tag{1.6.2}$$

First, a Lemma about the derivative of the activation function with respect to the encoder weight matrix.

Lemma 1.1. *Suppose $W_0 \in \mathbb{R}^{r\times n}, X \in \mathbb{R}^{n\times d}$ and σ is applied elementwise to the matrix W_0X, then*

$$\partial_{vec(W_0)} vec(\sigma(W_0X)) = diagvec(\sigma'(W_0X))[X^T \otimes I_r]. \tag{1.6.3}$$

Proof. I use the limit definition of the derivative to derive this result. Let $h \in \mathbb{R}^{r\times n}$ be arbitrary. In the limit as $h \to 0 \in \mathbb{R}^{r\times n}$ one can obtain

$$\mathrm{vec}(\sigma((W_0 + h)X) - \sigma(W_0X)) = \partial_{\mathrm{vec}(W_0)}(\mathrm{vec}(\sigma(W_0X)))\mathrm{vec}(h) \tag{1.6.4}$$

Expanding this term and noting that $\mathrm{vec}(\sigma(W_0X)) = \sigma(\mathrm{vec}(W_0X))$, as well as $\mathrm{vec}(A) \circ \mathrm{vec}(B) = \mathrm{diagvec}(A)\mathrm{vec}(B)$, one can obtain

$$\begin{aligned}
&\sigma(\mathrm{vec}((W_0 + h)X)) - \sigma(\mathrm{vec}(W_0X)) \\
=&\sigma(\mathrm{vec}(W_0X - hX)) - \sigma(\mathrm{vec}(W_0X)) \\
=&\mathrm{vec}(\sigma'(W_0X)) \circ \mathrm{vec}(hX) \\
=&\mathrm{diagvec}(\sigma'(W_0X))\mathrm{vec}(hX) \\
=&\mathrm{diagvec}(\sigma'(W_0X))[X^T \otimes I_r]\mathrm{vec}(h).
\end{aligned} \tag{1.6.5}$$

The result follows. □

Now, expressions for the gradients of the objective function $F(W_1, W_0)$.

Lemma 1.2. *The gradients of the objective function are given by*

$$\nabla_{W_1} F(W_1, W_0) = (W_1\sigma(W_0X) - Y)\sigma(W_0X)^T \tag{1.6.6}$$

$$\nabla_{W_0} F(W_1, W_0) = [\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))]X^T. \tag{1.6.7}$$

Proof. I derive in vectorized differential form, from which the matrix form derivatives can be extracted. First for the derivative with respect to D one can derive via the matrix partial differential only with respect to D:

$$\begin{aligned}\partial\text{misfit} &= -\partial\text{vec}(W_1\sigma(W_0X)) \\ &= -[\sigma(W_0X)^T \otimes I_n]\partial\text{vec}(W_1)\end{aligned} \tag{1.6.8}$$

Thus it follows that

$$\partial_{\text{vec}(W_1)}\text{misfit} = -[\sigma(W_0X)^T \otimes I_n] \tag{1.6.9}$$

The $\text{vec}(W_1)$ partial derivative is then:

$$\begin{aligned}&(\partial_{\text{vec}(W_1)}\text{misfit})^T\text{misfit} \\ =&[\sigma(W_0X)^T \otimes I_n]\text{vec}(W_1\sigma(W_0X) - Y) \\ =&\text{vec}((W_1\sigma(W_0X) - Y)\sigma(W_0X)^T).\end{aligned} \tag{1.6.10}$$

One can then express the matrix form partial derivative with respect to W_1 is:

$$\nabla_{W_1}F(W_1, W_0) = W_1\sigma(W_0X) - Y)\sigma(W_0X)^T. \tag{1.6.11}$$

For the partial derivative with respect to W_0, again start with the vectorized differential form.

$$\begin{aligned}\partial_{\text{vec}(W_0)}\text{misfit} &= -\partial_{\text{vec}(W_0)}\text{vec}(W_1\sigma(W_0X)) \\ &= -[I_d \otimes W_1]\partial_{\text{vec}(W_0)}\sigma(\text{vec}(W_0X))\end{aligned} \tag{1.6.12}$$

Applying Lemma 1.1 one can obtain:

$$\partial_{\text{vec}(W_0)}\text{misfit} = -[I_d \otimes W_1]\text{diagvec}(\sigma'(W_0X))[X^T \otimes I_r]. \tag{1.6.13}$$

The $\text{vec}(W_0)$ partial derivative is then:

$$\begin{aligned}&(\partial_{\text{vec}(W_0)}\text{misfit})^T\text{misfit} \\ =&[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))[I_d \otimes W_1^T]\text{vec}(W_1\sigma(W_0X) - Y) \\ =&[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))\text{vec}(W_1^T(W_1\sigma(W_0X) - Y)) \\ =&[X \otimes I_r]\text{vec}(\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))) \\ =&\text{vec}([\sigma'(W_0X) \circ (W_1^T(W_1\sigma(W_0X) - Y))]X^T),\end{aligned} \tag{1.6.14}$$

One can then obtain the matrix form partial derivative with respect to W_0 is:

$$\nabla_{W_0} F(W_1, W_0) = [\sigma'(W_0 X) \circ (W_1^T (W_1 \sigma(W_0 X) - Y))] X^T. \tag{1.6.15}$$

□

1.6.2 Derivation of Hessian

I now derive the four blocks of the Hessian matrix. I will proceed again by deriving partial differentials in vectorized form. In numerator layout one has

$$\begin{aligned} &\partial_{\mathrm{vec}(Y)} \partial_{\mathrm{vec}(X)} \left(\frac{1}{2} \mathrm{misfit}^T \mathrm{misfit} \right) = \\ &(\partial_{\mathrm{vec}(X)} \partial_{\mathrm{vec}(Y)} \mathrm{misfit})^T \mathrm{misfit} + (\partial_{\mathrm{vec}(X)} \mathrm{misfit})^T (\partial_{\mathrm{vec}(Y)} \mathrm{misfit}) \end{aligned} \tag{1.6.16}$$

The term involving only first partial derivatives of the misfit is the Gauss Newton portion which are already derived in section 1.6.1.

Lemma 1.3. *Gauss-Newton portions*

$$\begin{aligned} &(\partial_{vec(W_1)} misfit)^T (\partial_{vec(W_1)} misfit) \\ =&[\sigma(W_0 X) \sigma(W_0 X)^T \otimes I_n] && (1.6.17) \\ &(\partial_{vec(W_0)} misfit)^T (\partial_{vec(W_1)} misfit) \\ =&[X \otimes I_r] diagvec(\sigma'(W_0 X))[\sigma(W_0 X)^T \otimes W_1^T] && (1.6.18) \\ &(\partial_{vec(W_1)} misfit)^T (\partial_{vec(W_0)} misfit) \\ =&[\sigma(W_0 X) \otimes W_1] diagvec(\sigma'(W_0 X))[X^T \otimes I_r] && (1.6.19) \\ &(\partial_{vec(W_0)} misfit)^T (\partial_{vec(W_0)} misfit) \\ =&[X \otimes I_r] diagvec(\sigma'(W_0 X))[I_d \otimes W_1^T W_1] \cdots \\ &\qquad \cdots diagvec(\sigma'(W_0 X))[X^T \otimes I_r] && (1.6.20) \end{aligned}$$

Proof. This result follows from equations (1.6.9) and (1.6.13). □

I proceed by deriving the terms involving second partial derivatives of the misfit, by deriving their action on an arbitrary vector $Z \in \mathbb{R}^{n \times d}$. The

matrix $K^{(r,d)} \in \mathbb{R}^{dr \times rd}$ is the commutation (perfect shuffle) matrix satisfying the equality $K^{(r,d)}\text{vec}(V) = \text{vec}(V)^T$ for $V \in \mathbb{R}^{r \times d}$.

Lemma 1.4. *Non Gauss-Newton portions*

$$(\partial_{vec(W_1)}\partial_{vec(W_1)} misfit)^T misfit = 0 \tag{1.6.21}$$

$$\begin{aligned}&(\partial_{vec(W_0)}\partial_{vec(W_1)} misfit)^T misfit \\ =&[I_r \otimes (W_1\sigma(W_0X) - Y)]K^{(r,d)} dvec(\sigma'(W_0X))[X^T \otimes I_r]\end{aligned} \tag{1.6.22}$$

$$\begin{aligned}&(\partial_{vec(W_1)}\partial_{vec(W_0)} misfit)^T misfit \\ =&[X \otimes I_r] dec(\sigma'(W_0X))[(W_1\sigma(W_0X) - Y)^T \otimes I_r]K^{(n,r)}\end{aligned} \tag{1.6.23}$$

$$\begin{aligned}&(\partial_{vec(W_0)}\partial_{vec(W_0)} misfit)^T misfit \\ =&[X \otimes I_r] dvec\big([W_1^T(W_1\sigma(W_0X) - Y)] \\ &\qquad \circ \sigma''(W_0X)\big)[X^T \otimes I_r]\end{aligned} \tag{1.6.24}$$

Proof. Equation (1.6.21) follows from the fact that W_1 shows up linearly in the misfit. For the $W_0 - W_1$ block one has:

$$\begin{aligned}(\partial_{\text{vec}(W_1)}\text{misfit})^T\text{vec}(Z) &= -[\sigma(W_0X) \otimes I_n]\text{vec}(Z) \\ &= -\text{vec}(Z\sigma(W_0X)^T) \\ &= -[I_r \otimes Z]\text{vec}(\sigma(W_0X)^T) \\ &= -[I_r \otimes Z]K^{(r,d)}\text{vec}(\sigma(W_0X)).\end{aligned} \tag{1.6.25}$$

Equation (1.6.22) follows from taking a partial differential with respect to the vectorization of W_0, applying Lemma 1.1, and substituting the misfit for Z. For the $W_1 - W_0$ block one has:

$$\begin{aligned}&(\partial_{\text{vec}(W_0)}\text{misfit})^T\text{vec}(Z) \\ =&-[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))[I_d \otimes W_1^T]\text{vec}(Z) \\ =&-[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))\text{vec}(W_1^TZ) \\ =&-[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))[Z^T \otimes I_r]K^{(n,r)}\text{vec}(W_1).\end{aligned} \tag{1.6.26}$$

Equation (1.6.23) follows from taking a partial differential with respect to the vectorization of W_0 and subsituting the misfit for Z. Lastly for the $W_0 - W_0$ block one has:

$$\begin{aligned}
&(\partial_{\text{vec}(W_0)}\text{misfit})^T\text{vec}(Z) \\
=&-[X \otimes I_r]\text{diagvec}(\sigma'(W_0X))\text{vec}(W_1^TZ) \\
=&-[X \otimes I_r]\text{vec}(\sigma'(W_0X)) \circ \text{vec}(W_1^TZ) \\
=&-[X \otimes I_r]\text{vec}(W_1^TZ) \circ \text{vec}(\sigma'(W_0X)) \\
=&-[X \otimes I_r]\text{diagvec}(W_1^TZ)\text{vec}(\sigma'(W_0X)). \qquad (1.6.27)
\end{aligned}$$

Equation (1.6.24) follows from taking a partial differential with respect to the vectorization of W_0, applying Lemma 1.1, and substituting the misfit in for Z. □

1.7 Deep Dense Neural Network Gradient Derivation

The least squares loss function may be stated as:

$$\begin{aligned}
&F(W_0, W_1, \ldots, W_N) = \frac{1}{2}\text{misfit}^T\text{misfit} \\
&\text{misfit} = \text{vec}(Y - W_N\sigma_N(W_{N-1}\sigma_{N_1}(\cdots\sigma_1(W_0X)\cdots))). \qquad (1.7.1)
\end{aligned}$$

Numerator layout partial differentials are the same as in Equation (1.6.2). The partial derivatives require repeated application of the chain rule and Lemma 1.1, which can be stated:

$$\begin{aligned}
&\partial_{\text{vec}(W_j)}\text{vec}(\sigma_{j+2}(W_{j+1}\sigma_{j+1}(W_j\sigma_j(\cdots)))) \\
=&\text{dvec}(\sigma'_{j+2}(W_{j+1}\sigma_{j+1}(W_j\sigma_j(\cdots))))\cdots \\
&[I_d \otimes W_{j+1}]\partial_{\text{vec}(W_j)}\text{vec}(\sigma_{j+1}(W_j\sigma_j(\cdots))) \qquad (1.7.2)
\end{aligned}$$

Lemma 1.5. *Deep neural network gradients*

$$\begin{aligned}\nabla_{W_j} F(\mathbf{W}) = \Big[&\sigma'_{j+1}(W_j\sigma_j \cdots \sigma_1(W_0X)\cdots)\circ \\ &\big(W_{j+1}^T\big(\sigma'_{j+2}(W_{j+1}\sigma_j \cdots \sigma_1(W_0X)\cdots)\circ \\ &\cdots \circ \big(W_{N-1}^T\big(\sigma'_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)\big)\cdots \\ &\circ\big(W_N^T(W_N\sigma_N(W_{N-1}\cdots\sigma_1(W_0X)\cdots)-Y)\big)\cdots\big)\big)\Big] \\ &\sigma_j(W_j\cdots\sigma_1(W_0X))^T \end{aligned} \tag{1.7.3}$$

Proof. By iterative application of the chain rule (Equation (1.7.2)) one can derive the following

$$\begin{aligned}&\partial_{\mathrm{vec}(W_j)}\mathrm{misfit} = \\ -&[I_d \otimes W_N]\mathrm{dvec}(\sigma'_N(W_N\cdots\sigma_1(W_0X)\cdots))\cdots \\ &[I_d \otimes W_{j+1}]\mathrm{dvec}(\sigma'_{j+1}(W_j\cdots\sigma_1(W_0X)\cdots)) \\ &[\sigma_j(W_j\cdots\sigma_1(W_0X)\cdots)\otimes I_{r_j}].\end{aligned} \tag{1.7.4}$$

The result then follows from Equation (1.6.2) and properties of Kronecker and Hadamard products that are used in Appendix 1.6. □

Appendix 2

Scalable Methods for Learning Derivative Structure

In Chapter 2 I investigate the construction of dimension reducing surrogates for nonlinear parametric mappings with low dimensional structure. I investigate the ability to identify informed subspaces of the input and output of the map that are discretization dimension independent. This leads to an approach for parametric map surrogate construction that is scalable. Once the architecture for the surrogate model is set, the surrogate model is training via a stochastic optimization problem. In typical regression applications this involves a least-squares objective function that penalizes the square error of the surrogate model at points in the training dataset.

This strategy attempts to build a surrogate that converges to the true mapping $m \mapsto q$ in the L^2 sense. The empirical risk function can be thought of as a Monte Carlo approximation of the true integral with respect to the uncertain parameter probability distribution ν. In the limit that the Monte Carlo approximation converges to the true integral, and the surrogate model is able to drive the least squares error to zero, it has converged in L^2.

In many settings it is useful to construct a surrogate that approximates the mapping $m \mapsto q$ pointwise, but also well approximates its derivative $m \mapsto \nabla q(m)$. Regularity of the solution of a PDE u with respect to parameters m in the PDE (right hand sides, boundary conditions etc.) can be established in some contexts [137]. If the output quantity of interest is a continuous function (for example a linear operator) on the PDE solution (as is the case in examples I have explored), then in this context the mapping $m \mapsto q$ has at least a continuous first derivative. In other contexts for parametric maps,

derivatives can be assumed to exist.

In this setting I discuss extending the parametric surrogate strategy discussed in Chapter 2 to approximate derivatives of the parametric map as well. This is especially useful when many-query applications also use derivative information, examples of this include derivative based Monte Carlo [88], as well as many optimization problems [25].

2.1 Efficient Derivative Training

For the sake of simplicity, in this section I consider $m \in \mathbb{R}^{d_M}, q \in \mathbb{R}^{d_Q}$ to be real vectors. The uncertain parameter m has probability distribution ν. I assume that the mapping the function q is square integrable with respect to ν, that is, $q \in L^2(\mathbb{R}^{d_M}, \nu; \mathbb{R}^{d_Q})$. Also I assume its first derivative is also square integrable, i.e. $\nabla q \in L^2(\mathbb{R}^{d_M}, \nu; \mathbb{R}^{d_Q \times d_M})$.

Training first derivatives in this context seems like an expensive tasks, since it requires evaluation of a Frobenius norm to penalize the derivative of the map Given a neural network model $f(m, w)$, one seeks to minimize the expected risk of the first derivative of the mapping.

$$\int_{\mathbb{R}^{d_M}} \|\nabla_m f(m, w) - \nabla q(m)\|^2_{F(\mathbb{R}^{d_Q \times d_M})} d\nu(m) \tag{2.1.1}$$

One rarely has access to the true measure $\nu(m)$, and instead has to approximate the integral via Monte Carlo. The Monte Carlo approximation of the expected risk is known as the empirical. Given samples $\{(m_i, \nabla q(m_i))\}_{i=1}^N$, one can approximate the integral as

$$\frac{1}{N} \sum_{i=1}^{N} \|\nabla_m f(m_i, w) - \nabla q(m_i)\|^2_{F(\mathbb{R}^{d_Q \times d_M})}. \tag{2.1.2}$$

In applications of interest d_M is taken to be very large, making generation of training samples very expensive. Usually $d_Q < d_M$ and these Jacobians may have very fast spectral decay. In cases where the averaged Gauss Newton Hessian used in active subspace methods, or Fisher information have fast spectral decay we can reasonably assume the local Jacobians in this integral

have fast spectral decay as well. There should be a bound on the ranks of these operators since the Gauss Newton involves the "square" of the Jacobian operator, so when summed modes cannot cancel out.

The operator ∇q can reasonably be assumed to have low rank at points m_i. We can efficiently compute the rank r singular value decomposition at a given point in parameter space by its low rank SVD, $\nabla q(m_i) \approx U_r^{(i)} \Sigma_r^{(i)} V_r^{(i)T}$. We can then efficiently approximate the evaluation of the Frobenius norm

$$\|\nabla_m f(m_i, w) - \nabla q(m_i)\|^2_{F(\mathbb{R}^{d_Q \times d_M})} \approx \|U_r^{(i)T} \nabla_m f(m_i, w) V_r^{(i)} - \Sigma_r^{(i)}\|^2_{F(\mathbb{R}^{r \times r})}. \tag{2.1.3}$$

This approximation error can be bounded by the trailing singular values of the low rank Jacobian, and another constant involving a Lipshitz constant for the function f. Using this term in empirical risk minimization for the surrogate mapping $m \mapsto q$ enforces derivative in the locally informed subspaces of the Jacobian. This can be thought of as a form of regularization for the empirical risk minimization problem, specifically one that imposes local curvature constraints.

The computation of the low rank SVD of ∇q can be computed efficiently using adjoint methods discussed in Section 2.3. These methods require first a nonlinear solution of the mapping $m \mapsto q$ at a point m_i, and then all matrix-vector products after that linearization point involve only linear computations. The cost of these linearizations can be ammortized in computation for many different right hand sides. When the rank of the Jacobian at point m_i is small, the cost of generating Jacobian data can be made only marginally more than the cost of generating the zeroth order data. One ought to minimize the empirical risk for the Jacobian in conjunction with the empirical risk for the map $m \mapsto q$ itself anyways so it makes sense to use the same points for the evaluation of $q(m_i)$ as well as the linearization for $\nabla q(m_i)$ beyond just being computationally economical.

Training data for the low rank derivative, $\{(m_i, U_r^{(i)} \Sigma_r^{(i)} V_r^{(i)T})\}_{i=1}^N$ can be used to construct the active subspace projector used in Chapter 2. In the

Monte Carlo approximation of the action on a vector $\omega_M \in \mathbb{R}^{d_M}$ can be approximated as:

$$\frac{1}{N}\sum_{i=1}^{N} \nabla q(m_i)^T \nabla q(m_i)\omega_M \approx \frac{1}{N}\sum_{i=1}^{N} V_r^{(i)} \Sigma_r^{(i)2} V_r^{(i)T} \omega_M, \tag{2.1.4}$$

with an approximation error that is bounded by the largest trailing singular value across all N samples.

2.2 Approximation of derivative subspace

When one wants to resolve the dominant modes of the response surface for q, an optimal low rank approximation is given by the proper orthogonal decompostion, i.e. a low rank factorization of the matrix (approximated via Monte Carlo)

$$\frac{1}{N}\sum_{i=1}^{N} q(m_i)q(m_i)^T \approx \int_{\mathbb{R}^{d_M}} q(m)q(m)^T d\nu(m). \tag{2.2.1}$$

A low rank factorization of this matrix gives a basis that is effective in resolving the dominant span of q in $\mathbb{R}^{d_Q}$. When one wants to approximate dominant modes of the derivative of the response surface it makes sense to consider representation on a basis given by a low rank basis for the following matrix:

$$\mathbb{E}_\nu[\nabla q(m)\nabla q(m)^T] = \int_{\mathbb{R}^{d_M}} \nabla q(m)\nabla q(m)^T d\nu(m) \in \mathbb{R}^{d_Q \times d_Q}. \tag{2.2.2}$$

I refer to this matrix as the averaged inside out Gauss-Newton Hessian. The basis given by by the low rank vectors of this matrix is likely to span the dominant modes in which the derivative is active in the output representation. This matrix can be used to enrich a basis representation for a model that is meant to represent both q and ∇q. Like the averaged Gauss-Newton Hessian this matrix can be approximated via Monte Carlo integration, adjoint methods, and matrix-free randomized linear algebra. The action on a vector $\omega_Q \in \mathbb{R}^{d_Q}$ can be approximated as

$$\frac{1}{N}\sum_{i=1}^{N} \nabla q(m_i)\nabla q(m_i)^T \omega_Q. \tag{2.2.3}$$

An interesting question to explore, is how much of the dominant subspace for the averaged inside-out Gauss-Newton Hessian coincides with the dominant subspace of proper orthogonal decomposition. This is something I hope to explore in the future.

Appendix 3

Miscellaneous Proofs

3.1 Polynomials and Krylov Spaces

Note that generally for $x \in \mathcal{K}_m(A, v)$, we can express it as

$$\begin{aligned} x &= c_0 v + c_1 A v + \cdots + c_{i-1} A^{m-1} v \\ &= \underbrace{(c_0 I + c_1 A + \cdots + c_{m-1} A^{m-1})}_{p(A)} v = p(A)v \end{aligned} \tag{3.1.1}$$

for some $p \in \mathbb{P}_{m-1}$. When A is positive definite, the quadratic form $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ is bounded below. The minimizer of this quadratic form satisfies $Ax = b$. CG builds the approximation x_m in $\mathcal{K}_m(A, r_0)$ where $r_0 = b - Ax_0$ with the the property that the iterates minimize ϕ in each sequential Krylov subspace [48]. This is equivalent to the condition

$$x_m = \arg \min_{x \in x_0 + \mathcal{K}_m(A, r_0)} \|x_* - x\|_A^2 \tag{3.1.2}$$

where $\|y\|_A^2 = y^T Ay$ is the A-norm or energy-norm.

Proof of part of Theorem (3.4) (similar to results in [114] and [122]):

Proof. Given $x \in x_0 + \mathcal{K}_m(A, r_0)$ by (3.1.1) we may represent it as

$$x = x_0 + p(A)r_0 \tag{3.1.3}$$

for some $p \in \mathbb{P}_{m-1}$. Note that $r_0 = b - Ax_0 = Ax_* - Ax_0 = Ae_0$, we may then express $x_* - x$ as

$$x_* - x = e_0 - p(A)Ae_0 = \underbrace{(I - p(A)A)}_{q(A)} e_0 = q(A)e_0, \tag{3.1.4}$$

where $q \in \mathbb{P}_m$ is defined as

$$q(s) = 1 - sp(s) \quad \text{and} \quad q(0) = 1 \tag{3.1.5}$$

so $q \in \mathcal{Q}_m$. This establishes bijections between $\mathcal{K}_m(A, r_0), \mathbb{P}_{m-1}$ and $\mathcal{Q}_m$.From (3.1.2) we have

$$\|e_m\|_A^2 = \min_{x \in x_0 + \mathcal{K}_m(A, r_0)} \|x_* - x\|_A^2 = \min_{q \in \mathcal{Q}_m} \|q(A)e_0\|_A^2. \tag{3.1.6}$$

Representing e_0 on the eigenbasis for A ($A = U\Lambda U^T$) diagonalizes $q(A)$. We have then that

$$e_m = q(A)e_0 \tag{3.1.7a}$$

$$= \sum_{k=1}^{n} q(\lambda_k)(u_k^T e_0)u_k \tag{3.1.7b}$$

$$Ae_m = \sum_{k=1}^{n} \lambda_k q(\lambda_k)(u_k^T e_0)u_k \tag{3.1.7c}$$

$$e_m^T A e_m = \sum_{k=1}^{n} \lambda_k q(\lambda_k)^2 (u_k^T e_0)^2 \tag{3.1.7d}$$

□

The m^{th} iterate of MINRES is characterized as the unique point $x_m \in \mathcal{K}_m(A, b)$ such that ℓ^2 norm of the residual is minimal [100]:

$$x_m = \arg \min_{x \in \mathcal{K}_m(A,b)} \|b - Ax\|^2. \tag{3.1.8}$$

Proof of part of Theorem (3.4) (adapted from [7]):

Proof. Given $x \in \mathcal{K}_m(A, b)$, and employing (3.1.1) we can express the residual as

$$b - Ax = b - Ap(A)b = \underbrace{(I - Ap(A))}_{q(A)} b = q(A)b. \tag{3.1.9}$$

$q \in \mathbb{P}_m$ is defined as

$$q(s) = 1 - sp(s) \quad \text{and} \quad q(0) = 1 \tag{3.1.10}$$

so $q \in \mathcal{Q}_m$. This establishes bijections between $\mathcal{K}_m(A,b), \mathbb{P}_{m-1}$ and $\mathcal{Q}_m$. Applying (3.1.8) we have

$$\|b - Ax_m\|^2 = \min_{x \in \mathcal{K}_m(A,b)} \|b - Ax\|^2 = \min_{q \in \mathcal{Q}_m} \|q(A)b\|^2. \tag{3.1.11}$$

Expanding x and b on the eigenbasis for A ($A = U\Lambda U^T$) diagonalizes $q(A)$. The matrix polynomial optimization problem is then equivalent to a scalar polynomial optimization problem over the eigenvalues of A. This gives

$$\min_{q \in \mathcal{Q}_m} \|q(A)b\|^2 = \min_{q \in \mathcal{Q}_m} \sum_{k=1}^{n} q(\lambda_k)^2 (u_k^T b)^2 \tag{3.1.12}$$

□

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[3] Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the Hessian in deep neural networks. *arXiv preprint arXiv:1902.02366*, 2019.

[4] Alen Alexanderian, Noemi Petra, Georg Stadler, and Omar Ghattas. A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized ℓ_0-sparsification. *SIAM Journal on Scientific Computing*, 36(5):A2122–A2148, 2014.

[5] Alen Alexanderian, Noemi Petra, Georg Stadler, and Omar Ghattas. A fast and scalable method for A-optimal design of experiments for infinite-dimensional Bayesian nonlinear inverse problems. *SIAM Journal on Scientific Computing*, 38(1):A243–A272, 2016.

[6] Alen Alexanderian, Noemi Petra, Georg Stadler, and Omar Ghattas. Mean-variance risk-averse optimal control of systems governed

by PDEs with random parameter fields using quadratic approximations. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):1166–1192, 2017. arXiv preprint arXiv:1602.07592.

[7] Nick Alger. Data Scalable Hessian Preconditioning for Large-Scale PDE-constrained Inverse Problems, 2019.

[8] Animashree Anandkumar and Rong Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Conference on learning theory*, pages 81–102, 2016.

[9] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[10] Antonio Valerio Miceli Barone. Low-rank passthrough neural networks. *arXiv preprint arXiv:1603.03116*, 2016.

[11] O. Bashir, K. Willcox, O. Ghattas, B. van Bloemen Waanders, and J. Hill. Hessian-based model reduction for large-scale systems with initial condition inputs. *International Journal for Numerical Methods in Engineering*, 73:844–868, 2008.

[12] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[13] Dimitri P Bertsekas. Nonlinear Programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[14] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*, 2020.

[15] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2018.

[16] Jurgen Branke. Evolutionary algorithms for neural network design and training. Citeseer, 1995.

[17] Tan Bui-Thanh, Carsten Burstedde, Omar Ghattas, James Martin, Georg Stadler, and Lucas C. Wilcox. Extreme-scale UQ for Bayesian inverse problems governed by PDEs. In *SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012. Gordon Bell Prize finalist.

[18] Tan Bui-Thanh, Murali Damodaran, and Karen Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal*, 42(8):1505–1516, 2004.

[19] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part I: Inverse shape scattering of acoustic waves. *Inverse Problems*, 28(5):055001, 2012.

[20] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part II: Inverse medium scattering of acoustic waves. *Inverse Problems*, 28(5):055002, 2012.

[21] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part III: Inverse medium scattering of electromagnetic waves. *Inverse Problems and Imaging*, 7(4):1139–1155, 2013.

[22] Tan Bui-Thanh and Omar Ghattas. A scalable MAP solver for Bayesian inverse problems with Besov priors. *Inverse Problems and Imaging*, 9(1):27–54, 2015.

[23] Tan Bui-Thanh, Omar Ghattas, James Martin, and Georg Stadler. A computational framework for infinite-dimensional Bayesian inverse

problems Part I: The linearized case, with application to global seismic inversion. *SIAM Journal on Scientific Computing*, 35(6):A2494–A2523, 2013.

[24] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288, 2008.

[25] Tan Bui-Thanh, Karen Willcox, Omar Ghattas, and Bart van Bloemen Waanders. Goal-oriented, model-constrained optimization for reduction of large-scale systems. *Journal of Computational Physics*, 224(2):880–896, 2007.

[26] Chao Chen, Severin Reiz, Chenhan Yu, Hans-Joachim Bungartz, and George Biros. Fast evaluation and approximation of the gauss-newton hessian matrix for the multilayer perceptron. *arXiv preprint arXiv:1910.12184*, 2019.

[27] LONG CHEN. Johnson-Lindenstrauss transformation and random projection. 2015.

[28] Peng Chen and Omar Ghattas. Hessian-based sampling for high-dimensional model reduction. *International Journal for Uncertainty Quantification*, 9(2), 2019.

[29] Peng Chen, Alfio Quarteroni, and Gianluigi Rozza. A weighted reduced basis method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 51(6):3163–3185, 2013.

[30] Peng Chen, Alfio Quarteroni, and Gianluigi Rozza. Reduced basis methods for uncertainty quantification. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):813–869, 2017.

[31] Peng Chen and Christoph Schwab. Model order reduction methods in computational uncertainty quantification. *Handbook of uncertainty quantification*, pages 1–53, 2016.

[32] Peng Chen, Umberto Villa, and Omar Ghattas. Hessian-based adaptive sparse quadrature for infinite-dimensional bayesian inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 327:147–172, 2017.

[33] Peng Chen, Umberto Villa, and Omar Ghattas. Taylor approximation for pde-constrained optimization under uncertainty: Application to turbulent jet flow. *PAMM*, 18(1):e201800466, 2018.

[34] Peng Chen, Umberto Villa, and Omar Ghattas. Taylor approximation and variance reduction for pde-constrained optimal control under uncertainty. *Journal of Computational Physics*, 385:163–186, 2019.

[35] Peng Chen, Keyi Wu, Joshua Chen, Tom O'Leary-Roseberry, and Omar Ghattas. Projected stein variational newton: A fast and scalable bayesian inference method in high dimensions. In *Advances in Neural Information Processing Systems*, pages 15104–15113, 2019.

[36] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[37] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[38] Benjamin Crestel, Alen Alexanderian, Georg Stadler, and Omar Ghattas. A-optimal encoding weights for nonlinear inverse problems, with application to the Helmholtz inverse problem. *Inverse Problems*, 33(7):074008, 2017.

[39] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[40] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in neural information processing systems*, pages 1504–1512, 2015.

[41] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[42] Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical analysis*, 19(2):400–408, 1982.

[43] Stanley C Eisenstat and Homer F Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[44] Heinz W. Engl, Martin Hanke, and Andreas Neubauer. *Regularization of Inverse Problems*, volume 375 of *Mathematics and Its Applications*. Springer Netherlands, 1996.

[45] M. A. Erdogdu and A. Montanari. Convergence Rates of Sub-sampled Newton Methods. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3052–3060. Curran Associates, Inc., 2015.

[46] Dave Fernandes. SaddleFreeOptimizer. `https://https://github.com/dave-fernandes/SaddleFreeOptimizer/`, 2019.

[47] Pearl H. Flath, Lucas C. Wilcox, Volkan Akçelik, Judy Hill, Bart van Bloemen Waanders, and Omar Ghattas. Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations. *SIAM Journal on Scientific Computing*, 33(1):407–432, 2011.

[48] David Chin-Lung Fong and Michael Saunders. CG versus Minres: An empirical comparison. *Sultan Qaboos University Journal for Science [SQUJS]*, 17(1):44–62, 2012.

[49] GAEL Forget, J-M Campin, Patrick Heimbach, Christopher N Hill, Rui M Ponte, and Carl Wunsch. Ecco version 4: An integrated framework for non-linear inverse modeling and global ocean state estimation. 2015.

[50] Stefania Fresca, Luca Dede, and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *arXiv preprint arXiv:2001.04001*, 2020.

[51] Ichiro Fukumori, Patrick Heimbach, Rui M Ponte, and Carl Wunsch. A dynamically consistent, multivariable ocean climatology. *Bulletin of the American Meteorological Society*, 99(10):2107–2128, 2018.

[52] Isha Garg, Priyadarshini Panda, and Kaushik Roy. A low effort approach to structured cnn design using pca. *IEEE Access*, 2019.

[53] Matilde Gargiani, Andrea Zanelli, Moritz Diehl, and Frank Hutter. On the promise of the stochastic generalized Gauss-Newton method for training dnns. *arXiv preprint arXiv:2006.02409*, 2020.

[54] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

[55] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An Investigation into Neural Net Optimization via Hessian Eigenvalue Density. *arXiv preprint arXiv:1901.10159*, 2019.

[56] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization*. Academic press, 1981.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[58] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[59] M. Hanke. *Conjugate Gradient Type Methods for Ill–Posed Problems*. Pitman Research Notes in Mathematics, Vol 327. Longman Scientific & Technical, Essex, 1995.

[60] P. C. Hansen. *Rank Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, 1998.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[62] Patrick Heimbach, Ichiro Fukumori, Christopher N Hill, Rui M Ponte, Detlef Stammer, Carl Wunsch, Jean-Michel Campin, Bruce D Cornuelle, Ian Fenty, Gaël Forget, et al. Putting it all together: Adding value to the global ocean and climate observing systems with complete self-consistent ocean state and parameter estimates. 2019.

[63] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[64] Tobin Isaac, Noemi Petra, Georg Stadler, and Omar Ghattas. Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet. *Journal of Computational Physics*, 296:348–368, September 2015.

[65] Tobin Isaac, Noemi Petra, Georg Stadler, and Omar Ghattas. Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the antarctic ice sheet. *Journal of Computational Physics*, 296:348–368, 2015.

[66] Tobin Isaac, Georg Stadler, and Omar Ghattas. Solution of nonlinear stokes equations discretized by high-order finite elements on nonconforming and anisotropic meshes, with application to ice sheet dynamics. *SIAM Journal on Scientific Computing*, 37(6):B804–B833, 2015.

[67] Prateek Jain, Chi Jin, Sham M Kakade, and Praneeth Netrapalli. Computing matrix squareroot via non convex local search. arxiv preprint. *arXiv preprint arXiv:1507.05854*, 2015.

[68] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

[69] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017.

[70] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[71] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[72] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). 2010.

[73] Jakob Kruse, Gianluca Detommaso, Robert Scheichl, and Ullrich Köthe. Hint: Hierarchical invertible neural transport for density estimation and bayesian inference. *arXiv preprint arXiv:1905.10687*, 2019.

[74] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[75] Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*, 2017.

[76] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257, 2016.

[77] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[78] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[79] Chad Lieberman, Karen Willcox, and Omar Ghattas. Parameter and state model reduction for large-scale statistical inverse problems. *SIAM Journal on Scientific Computing*, 32(5):2523–2542, 2010.

[80] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[81] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[82] Andrea Manzoni, Federico Negri, and Alfio Quarteroni. Dimensionality reduction of parameter-dependent problems through proper orthogonal decomposition. *Annals of Mathematical Sciences and Applications*, 1(2):341–377, 2016.

[83] J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

[84] J. Martens and I. Sutskever. *Training Deep and Recurrent Networks with Hessian-Free Optimization*, pages 479–535. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[85] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

[86] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

[87] James Martin, Lucas C. Wilcox, Carsten Burstedde, and Omar Ghattas. A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion. *SIAM Journal on Scientific Computing*, 34(3):A1460–A1487, 2012.

[88] James Martin, Lucas C Wilcox, Carsten Burstedde, and Omar Ghattas. A stochastic Newton mcmc method for large-scale statistical inverse

problems with application to seismic inversion. *SIAM Journal on Scientific Computing*, 34(3):A1460–A1487, 2012.

[89] P. G. Martinsson and J. Tropp. Randomized Numerical Linear Algebra: Foundations & Algorithms. *arXiv preprint arXiv:2002.01387*, 2020.

[90] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *arXiv preprint arXiv:1903.00104*, 2019.

[91] Craig Michoski, Miloš Milosavljević, Todd Oliver, and David R Hatch. Solving differential equations using deep neural networks. *Neurocomputing*, 2020.

[92] Jorge J Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[93] Katta G Murty and Santosh N Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[94] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[95] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[96] Thomas O'Leary-Roseberry, Nick Alger, and Omar Ghattas. Inexact Newton methods for stochastic non-convex optimization with applications to neural network training. *arXiv preprint arXiv:1905.06738*, 2019.

[97] Thomas O'Leary-Roseberry, Nick Alger, and Omar Ghattas. Low Rank Saddle Free Newton: Algorithm and Analysis. *arXiv preprint arXiv:2002.02881*, 2020.

[98] Thomas O'Leary-Roseberry and Omar Ghattas. Ill-posedness and optimization geometry for nonlinear neural network training. *arXiv preprint arXiv:2002.02882*, 2020.

[99] Carlos E Orozco and ON Ghattas. Massively parallel aerodynamic shape optimization. *Computing Systems in Engineering*, 3(1-4):311–320, 1992.

[100] Chris C Paige, Beresford N Parlett, and Henk A Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical linear algebra with applications*, 2(2):115–133, 1995.

[101] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[102] Mario Teixeira Parente, Jonas Wallin, Barbara Wohlmuth, et al. Generalized bounds for active subspaces. *Electronic Journal of Statistics*, 14(1):917–943, 2020.

[103] Santiago Paternain, Aryan Mokhtari, and Alejandro Ribeiro. A Newton-Based Method for Nonconvex Optimization with Fast Evasion of Saddle Points. *SIAM Journal on Optimization*, 29(1):343–368, 2019.

[104] Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994.

[105] Noemi Petra, James Martin, Georg Stadler, and Omar Ghattas. A computational framework for infinite-dimensional Bayesian inverse problems: Part II. Stochastic Newton MCMC with application to ice sheet

inverse problems. *SIAM Journal on Scientific Computing*, 36(4):A1525–A1555, 2014.

[106] T Poggio and Q Liao. Theory i: Deep networks and the curse of dimensionality. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 66(6), 2018.

[107] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[108] Maziar Raissi and George Karniadakis. Deep multi-fidelity Gaussian processes. *arXiv preprint arXiv:1604.07484*, 2016.

[109] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[110] Sashank J Reddi, Manzil Zaheer, Suvrit Sra, Barnabas Poczos, Francis Bach, Ruslan Salakhutdinov, and Alexander J Smola. A generic approach for escaping saddle points. *arXiv preprint arXiv:1709.01434*, 2017.

[111] F. Roosta-Khorasani and M. W. Mahoney. Sub-sampled Newton methods i: globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016.

[112] F. Roosta-Khorasani and M. W. Mahoney. Sub-sampled Newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016.

[113] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[114] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[115] Sirpa Saarinen, Randall Bramley, and George Cybenko. Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714, 1993.

[116] Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. *arXiv preprint arXiv:1712.08968*, 2017.

[117] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

[118] Otmar Scherzer. The use of morozov's discrepancy principle for tikhonov regularization for solving nonlinear ill-posed problems. *Computing*, 51(1):45–60, 1993.

[119] Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

[120] Christoph Schwab and Radu Alexandru Todor. Karhunen–Loève approximation of random fields by generalized fast multipole methods. *Journal of Computational Physics*, 217(1):100–122, 2006.

[121] Christoph Schwab and Jakob Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01):19–55, 2019.

[122] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[123] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020.

[124] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504. ACM, 2017.

[125] Ju Sun, Qing Qu, and John Wright. When are nonconvex problems not scary? *arXiv preprint arXiv:1510.06096*, 2015.

[126] Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 2379–2383. IEEE, 2016.

[127] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Willcox. Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, 179:704–717, 2019.

[128] Andrei Nikolajevits Tihonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math.*, 4:1035–1038, 1963.

[129] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[130] Rohit Tripathy and Ilias Bilionis. Deep active subspaces: A scalable method for high-dimensional uncertainty propagation. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[131] Umberto Villa, Noemi Petra, and Omar Ghattas. hIPPYlib: An extensible software framework for large-scale inverse problems. *The Journal of Open Source Software*, 3:940, 2018.

[132] Karen Willcox and Jaime Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA journal*, 40(11):2323–2330, 2002.

[133] Carl Wunsch and Patrick Heimbach. Practical global oceanic state estimation. *Physica D: Nonlinear Phenomena*, 230(1-2):197–208, 2007.

[134] P. Xu, J. Yang, F. Roosta-Khorasani, C. Ré, and M. W. Mahoney. Sub-sampled Newton Methods with Non-uniform Sampling. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3000–3008. Curran Associates, Inc., 2016.

[135] Peng Xu, Farbod Roosta-Khorasan, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827*, 2017.

[136] Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *arXiv preprint arXiv:1708.07164*, 2017.

[137] xxx. *Numerical Approximation of High-Dimensional PDEs : Applications in Uncertainty Quantification*. preprint, 2020.

[138] Xin Yao. Evolutionary artificial neural networks. *International journal of neural systems*, 4(03):203–222, 1993.

[139] Olivier Zahm, Paul G Constantine, Clementine Prieur, and Youssef M Marzouk. Gradient-based dimension reduction of multivariate vector-valued functions. *SIAM Journal on Scientific Computing*, 42(1):A534–A558, 2020.

[140] Zhihui Zhu, Daniel Soudry, Yonina C Eldar, and Michael B Wakin. The global optimization geometry of shallow linear neural networks. *Journal of Mathematical Imaging and Vision*, pages 1–14, 2018.